

Oracle® Cloud

Using Oracle Process Cloud Service



E68292-14
August 2018



Oracle Cloud Using Oracle Process Cloud Service,

E68292-14

Copyright © 2015, 2018, Oracle and/or its affiliates. All rights reserved.

Primary Author: Oracle Corporation

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

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

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

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

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

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

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

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

Contents

Preface

Audience	xvii
Documentation Accessibility	xvii
Related Resources	xvii
Conventions	xvii

1 Getting Started with Process

What Is Process?	1-1
Before You Begin with Process	1-2
How to Begin with Oracle Process Cloud Service Subscriptions	1-2
About Process Roles and Users	1-2
Signing In to Oracle Process Cloud Service	1-4
Quick Tour of the Home Page	1-5
Quick Tour of the Composer Page	1-6

Part I End User Tasks

2 Working on Tasks

What can I do with Process?	2-1
How do I start an application?	2-2
Completing Tasks	2-2
How do I work on tasks?	2-3
How do I find, filter, and sort my tasks?	2-4
Filtering the Task List	2-6
Sorting the Task List	2-7
How do I add a document to a task?	2-7
How can I manage documents in folders?	2-8
Can I reassign my tasks to someone else?	2-9
Can I collaborate with others about a task?	2-9
Setting Preferences	2-10

How do I set my out-of-office preferences?	2-10
Can I configure accessibility options?	2-11
How do I use the mobile app?	2-11

3 General Troubleshooting

I can't sign in	3-1
I don't see any applications to start	3-1
I don't see any assigned tasks	3-1

Part II Developer Tasks

4 Developer Basics

QuickStart Apps, Applications, and QuickStart Masters	4-1
Creating Your First Application (a Quick Start)	4-2
Create from a QuickStart App	4-2
Customize the QuickStart App	4-4
Test Activate the QuickStart App	4-5
Try It in Test Mode	4-6
Activate the QuickStart App	4-7
Personalizing and Promoting QuickStart Apps to the Gallery	4-8
Convert an Application to a QuickStart Master	4-8
Configure How the Application Looks in the Gallery	4-10
Decide What Settings Users Can Customize	4-11
Promote the Application to the Gallery	4-12
Learning More about the Components in an Application	4-13
Managing the Development Life Cycle	4-15
Setting Preferences for the Design-Time Environment	4-16

5 Creating and Managing Applications

What You Can Do on the Application Home Tab	5-1
About Application Sharing and Collaboration	5-5
Managing Spaces for Sharing and Collaboration	5-6
About QuickStart Apps	5-7
Creating a New Application	5-8
Editing Application Properties	5-9
Validating an Application	5-11
Saving and Publishing Changes to an Application	5-11
Playing Processes and Testing Applications	5-12

Mapping Process Roles to Users in Your Organization	5-12
About Testing Processes Using the Application Player	5-14
Testing a Business Process	5-15
About Testing an Application's Activation	5-18
Testing an Application's Activation	5-19
Customizing Titles	5-21
Customizing Instance Titles Displayed in Runtime	5-21
Customizing Task Titles Displayed in Runtime	5-22
Working with Application Snapshots	5-23
Viewing the Change History of an Application	5-24
Defining Application Roles	5-24
Defining Business Parameters	5-25
Localizing Applications	5-26
Importing and Exporting Applications and Snapshots	5-28
Importing an Application from your Local File System	5-29
Exporting an Application	5-29
Exporting Applications or Snapshots from the Application Home Tab	5-29
Documenting Applications	5-30
Creating Process-Level Documentation	5-30
Creating Application-Level Documentation	5-32
Creating Activity-Level Documentation	5-33
Generating Application Reports	5-36

6 Developing Structured Processes

About Business Processes	6-1
What You Can Do Using the Process Editor	6-5
Typical Workflow for Creating a Business Process	6-8
Creating a Business Process	6-8
Deleting a Business Process	6-10
Importing a Process Created Externally	6-11
Working with Process Roles and Swimlanes	6-11
Adding and Editing Swimlanes	6-13
Working with Elements	6-13
Adding Elements	6-15
Editing an Element's Properties	6-16
Defining Process Start and End	6-16
About the None Start Event	6-18
About the Form Start Event	6-18
About the Document Start Event	6-18
About the Folder Start Event	6-19

About the Message Start Event	6-19
About the None End Event	6-20
About the Message End Event	6-20
About the Error End Event	6-21
About the Terminate End Event	6-21
Adding Human Interaction	6-21
Connecting to Other Processes and Services	6-22
Controlling Process Flow Using Gateways	6-24
Exclusive Gateway: Take Only One Path (Data Condition)	6-24
Inclusive Gateway: Take One or More Paths	6-25
Parallel Gateway: Take All Paths Simultaneously	6-26
Event-Based Gateway: Take Only One Path (Event Occurrence)	6-28
Creating a Gateway	6-29
Working with Decisions	6-31
Working with Rules	6-31
Working with Sequence Flows	6-32
About Unconditional Sequence Flows	6-33
About Conditional Sequence Flows	6-33
About Default Sequence Flows	6-34
Working with Intermediate Events	6-34
About the Timer Catch Event	6-34
About the Error Boundary Event	6-35
Working with Draft Processes	6-36
About the Abstract Flow Element	6-36
Working with Subprocesses	6-36
About Reusable Processes (Reusable Subprocesses)	6-37
About Embedded Subprocesses (Inline Subprocesses)	6-37
About Event Subprocesses (Event Handlers)	6-38
Configuring Multi-Instance Markers in Subprocesses	6-38
Communicating Between Processes	6-41
Using Send and Receive to Communicate Between Processes	6-42
Using Message Throw and Catch to Communicate Between Processes	6-43
Defining Input or Output Arguments	6-44
Defining Input and Output Arguments for a Child Process	6-45
Calling Another Process	6-46
Using Correlations to Communicate Between Processes	6-47
Correlations Lead to Process Communication	6-47
Components of a Correlation	6-47
Defining Correlation Keys and Properties	6-48
Using Your Correlations in a Process	6-48
Working with Human Tasks	6-51

Typical Workflow for Creating a Human Task	6-51
Creating Submit and Approval Tasks	6-51
Configuring Human Tasks	6-52
Assigning Human Tasks	6-53
Using Forms to Display Task Information	6-54
Defining an Approval Pattern	6-55
Configuring the Title and Task Summary	6-56
Configuring the Due Date and Priority	6-56
Bypassing the Approval Chain	6-57
Configuring Reminders	6-58
Configuring Task Expiration, Renewal, or Escalation	6-59
Overriding Documents Folder Access	6-60
Changing Conversation Settings for a Human Task	6-62
Customizing Notification Emails for Human Tasks	6-62
About Notification Email and Templates	6-63
Managing Email Templates	6-64
Configuring Email Templates	6-65

7 Creating Web Forms

Working in the Web Forms Editor	7-1
Ready to create a web form?	7-3
Create a Simple Application	7-3
Create a Web Form	7-4
Add and Configure Controls	7-5
Add Another Presentation	7-8
Change the Form's Stylesheet	7-9
Preview the Form	7-10
Use Forms and Presentations in a Process	7-11
Define a Control's Behavior	7-12
Try the Form in Runtime	7-14
Explore Advanced Form Options	7-15
Positioning Controls on Forms	7-15
Working with Presentations	7-17
Binding Form Data with Controls	7-19
Creating a Web Form Based on a Business Type	7-20
Working with Stylesheets and Styling	7-20
Styling Properties	7-21
Customizing Web Forms Using CSS	7-23
Styling the Form Container	7-23
Styling Form Controls	7-24

Form Reference Styling	7-31
Styling for Imported Business Types	7-32
Managing Style Dynamically	7-32
Some Useful Styling Tips	7-35
Configuring Basic Controls	7-35
Configuring Text Input and Area Fields	7-35
Configuring Buttons	7-36
Configuring Drop-down Select Fields	7-37
Configuring Check Lists and Check Boxes	7-38
Configuring Radio Buttons	7-39
Configuring Number Fields	7-40
Configuring Date and Time Fields	7-41
Configuring Email Fields	7-42
Configuring Web Address URL Fields	7-43
Configuring Message Fields	7-43
Configuring Links	7-44
Configuring Advanced Controls	7-45
Configuring Currency (Money) Fields	7-46
Configuring Phone Number Fields	7-47
Including Images	7-47
Including Videos	7-48
Configuring Identity Browser Fields	7-49
Placing Controls in Panels, Sections, or Tabs	7-51
Configuring Static and Dynamic List of Values Fields	7-52
Configuring Repeatable Sections	7-53
Configuring Tables	7-55
Creating Computed Controls	7-56
Localizing Web Forms	7-58
Previewing Forms and Their Payload	7-62
Adding Dynamic Behaviors to a Form	7-62
Configuring Events	7-63
Specifying Actions	7-65
Specifying Conditions	7-67
Specifying Functions	7-70
Executing REST Connector Calls in Events	7-73
Populating Controls Using REST Calls	7-76
Linking and Refreshing List of Value Fields	7-78
Reusing Forms	7-78

8 Creating Basic Forms

About Basic Forms	8-1
Typical Workflow Using Form-First Design	8-2
Typical Workflow for Using Data-First Design	8-3
What You Can Do Using the Basic Forms Designer	8-4
Creating and Editing Basic Forms	8-9
Creating a Basic Form from the Application Home Tab	8-9
Creating a Basic Form from the Process Editor	8-10
Editing Basic Form Properties	8-11
Working with Basic Form Controls	8-12
Adding Input Control Fields	8-12
Adding Selection Control Fields	8-16
Adding Group Controls	8-17
Adding Other Controls	8-21
Editing Basic Form Control Properties	8-23
Adding a Data Source or Business Object to a Basic Form	8-24
Editing Basic Form Controls Generated by a Data Source or Business Object	8-26
Showing Which Basic Form Controls Were Created by a Data Source or Business Object	8-27
Previewing a Basic Form	8-27
Saving Basic Form Data	8-28
Configuring Basic Form Snapshots	8-28
Creating Basic Form Rules	8-32
JavaScript Syntax for Basic Form Rules	8-32
Creating a Basic Form Rule	8-38
Using Dynamic Content	8-38
Using Dynamic Validation	8-40
Using Built-In Data	8-41
Using Built-In Methods	8-41
Working with REST Calls with Custom Headers	8-42
Managing Header Sets Used in Basic Form Requests	8-42
Testing and Debugging a Basic Form Rule	8-43

9 Managing Application Data

About Managing Data	9-1
Defining Data	9-1
Typical Workflow for Defining the Data Used Within a Business Process	9-2
What You Can Do on the Business Types Page	9-2
Working with Business Objects	9-5
Creating a Business Object	9-7

Importing a Business Object from XML	9-8
Importing a Business Object from JSON	9-8
Importing XML Schema Files	9-11
Uploading a New Version of an XML Schema File	9-12
Promoting Schema Types and Elements to Business Objects	9-13
Managing Business Types	9-14
Working with Data Objects	9-16
Creating a Data Object	9-17
Editing or Deleting a Data Object	9-17
Creating a Business Exception	9-18
Creating an Enum Object	9-19
Working with Business Indicators	9-20
Creating Business Indicators	9-22
Associating and Manipulating Data	9-25
Working with Expressions	9-25
Configuring Data Association	9-34
Data Association Tips	9-36
Data Association for Enums	9-36
Working with Transformations	9-37
Creating, Applying, and Editing Transformations	9-38
Creating Transformations Between Enum Items	9-41

10 Creating Decisions

How do decisions work in process applications?	10-1
Ready to create a decision model?	10-2
Create a Simple Application	10-2
Create a Decision Model	10-4
Add Decisions	10-5
Define Input	10-6
Model Decision Logic	10-8
Test Your Decisions	10-12
Create a Service	10-13
Deploy a Decision Snapshot	10-14
Use the Decision in Your Application	10-15
Map Data to and from the Decision	10-16
Try Your Decision in Runtime	10-18
Working with Decision Models	10-18
Creating a Decision Model	10-19
Adding and Ordering Decisions	10-22
Understanding FEEL (Friendly Enough Expression Language)	10-23

Data Types	10-23
Grammar Rules	10-24
Built-In Functions	10-25
Conversion Functions	10-25
Boolean Functions	10-26
String Functions	10-26
List Functions	10-27
Numeric Functions	10-28
Working with Date, Time, and Duration Functions	10-28
Conversion Operations	10-29
Arithmetic Operations	10-29
Comparison Operations	10-30
Defining Decision Input and Type	10-30
Defining a Complex Data Type	10-31
Creating Input Data	10-32
Modeling Decision Logic	10-33
Using Empty Logic Decisions	10-33
Using Decision Tables	10-33
Defining Decision Table Input	10-35
Defining Decision Table Output	10-36
Configuring Rules	10-37
Configuring a Hit Policy	10-37
Using Expressions	10-44
Using If-Then-Else Statements	10-45
Using Functions	10-46
Using Contexts	10-48
Using Lists	10-50
Using Relations	10-52
Testing Decisions	10-54
Exposing Decisions as Services	10-54
Deploying and Managing Decision Model Snapshots	10-54
Adding Decisions to Applications and Processes	10-56
Importing and Exporting Decision Models	10-56
Importing a Decision Model	10-57
Exporting a Decision Model	10-57

11 Creating Business Rules

About Business Rules	11-1
What You Can Do Using the Business Rules Editor	11-3
Typical Workflow for Creating Business Rules	11-4

Creating a Decision in Business Rules	11-5
Creating a Value Set	11-6
Creating a Range Value Set	11-8
Creating Global Variables	11-9
Creating an If/Then Rule in Business Rules	11-10
Creating a Decision Table in Business Rules	11-12
Assigning a Business Rule to a Process Component	11-14
Typical Workflow for Using Advanced Business Rule Features	11-16
Using Advanced Conditions	11-17
Using Advanced Actions	11-20
Editing Rule and Decision Table Properties	11-21
Adding Conflict Resolutions to Decision Tables	11-22
Editing Decision Properties in Business Rules	11-25

12 Integrating Documents and Conversations

Why should I integrate documents and conversations?	12-1
How do I integrate with Oracle Content and Experience Cloud?	12-2
Access Requirements for a Successful Integration	12-2
Using Documents or Attachments in a Process Application	12-3
Configuring Settings in Oracle Content and Experience Cloud	12-4
Configuring Documents Settings in Oracle Process Cloud Service	12-4
Enabling or Disabling Documents and Conversations	12-5
Access Issues and Configuration Changes	12-6
Roles and Responsibilities	12-6
Quick Tour of the Documents Page	12-7
Editing Properties of the Documents Root Folders	12-10
Defining Folders and Documents	12-11
Creating a Document- or Folder-Initiated Process	12-12
Using REST API Calls to Instantiate a Process	12-16
Getting Properties for Documents, Folders, and Conversations	12-18
Editing Conversation Properties	12-20
Sample Use Case for Documents and Process	12-22

13 Integrating with Applications and Services

What You Can Do on the Integrations Page	13-1
Working with Integration Cloud Service Integrations	13-3
About ICS Integrations	13-3
Configuring ICS Integrations	13-4
Editing ICS Integrations	13-5

Creating REST and Web Service Connectors	13-7
About REST and Web Services	13-7
Creating a REST Connector	13-8
Working with Web Service Definition Files	13-13
Preparing Local WSDL Files Containing Remote References	13-15
Creating a Web Service Connector	13-15
Applying Message Security to Integrations	13-18
Invoking ICS Integrations, REST, and Web Services	13-19
Embedding Process UI Components in Other Applications	13-19
Process UI Components Available for Embedding	13-20
Before You Embed Process UI Components	13-21
Downloading Process UI Component Files	13-21
Experimenting with Process UI Components	13-22
Consuming the Process UI Component	13-25

14 Troubleshooting Application Development

Troubleshooting Application Import	14-1
Troubleshooting Business Processes	14-2
Troubleshooting Basic Form Rules	14-3
Troubleshooting Data Association	14-4
Troubleshooting Business Rules	14-5
Troubleshooting Application Playing	14-5

Part III Administrator Tasks

15 Administrator Basics

About Administrator Privileges	15-1
Configuring Federated SSO and Authentication	15-1
Using OAuth with REST APIs	15-3
Configuring OAuth Resources and Clients	15-4
Obtaining a Client Access Token	15-4

16 Managing the Runtime Environment

Task Overview for Administering the Runtime Environment	16-1
Assigning and Managing Roles	16-1
Monitoring and Adjusting Processes	16-2
Viewing Alerts	16-3
Tracking Process Instances	16-3

Altering the Flow of a Process Instance	16-3
Monitoring Key Metrics in Dashboards	16-4
Viewing and Resending Email Notifications	16-5
Creating and Viewing Business Analytics Dashboards	16-6
Selecting System Indicators	16-9
Creating Analytics with Oracle Business Intelligence Cloud Service	16-10
How do I integrate with Oracle Business Intelligence Cloud Service?	16-11
Configuring BI Cloud Service Settings in Process	16-11
Creating Charts and Graphs in Oracle BI Cloud Service	16-12
Configuring Application Settings	16-13
Configuring Oracle Integration Cloud Service	16-14
Configuring Audit and Log Levels	16-14
Enabling Email Notifications	16-16
Configuring Credentials for Web Services	16-16
Configuring Credentials for Basic Form REST Services	16-17
Managing Security Certificates during Runtime	16-18
Resolving Certificate Validation Exceptions	16-19
Customizing the User Interface	16-19
Configuring Oracle Content and Experience Cloud	16-20
Archiving and Purging Data	16-20
About Archive and Purge	16-21
Archiving On Demand	16-22
Scheduling Instances Archive and Purge	16-23
Scheduling Instance Purge Only	16-25
Purging Transient Data	16-25
Scheduling Analytics Data Archive and Purge	16-26
Scheduling Analytics Data Purge Only	16-27
Viewing Archive Requests	16-28
Troubleshooting the Runtime Administration	16-29

17 Managing the Design-Time Environment

Administering Spaces, Applications, and the Player	17-1
Administering Spaces	17-1
Unlocking and Deleting Applications	17-2
Deleting QuickStart Apps from the Gallery	17-2
Enabling the Application Player	17-3
Managing Environments and Activating Applications	17-4
Configuring the Activation Permissions	17-5
Activating Applications	17-5
Managing Active Applications	17-10

Configuring a Remote Server	17-12
Managing Security Certificates during Design Time	17-13

Part IV Appendices

A Basic Form and Basic Form Control Property Reference

Basic Form Properties	A-1
Properties on the Settings Tab for Basic Forms	A-1
Properties on the Style Tab for Basic Forms	A-2
Basic Form Control Properties	A-3
Properties on the Settings Tab for Basic Form Controls	A-3
Properties on the Style Tab for Basic Form Controls	A-10

B Basic Form Rule Examples

Calculate a Total	B-2
Show/Hide a Billing Address	B-3
Show/Hide a Message	B-3
Enable/Disable a Question	B-3
Compute Subtotals for Repeating Items	B-4
Compute an Invoice Total	B-4
Textarea Max Length	B-5
Textarea Newline and Break	B-5
Drop-Down Options	B-6
Finding a Selected Options Index	B-6
Synchronized Selects	B-7
Clearing Drop-Down Options	B-7
Default Option	B-8
Check Box Options - Assigning Color to Check Box Choices	B-8
Check Box Options - Making a Control Visible/Invisible Based on Check Box Choices	B-9
Check Box Initialization	B-10
Displaying Selected Check Box Labels	B-10
Repeating Check Boxes	B-10
Display a Message Control Inside a Control	B-11
String Concatenation	B-11
Visible/Invisible	B-12
Visible/Invisible Section	B-12
Select Tab	B-13
Next Tab	B-13

Expand/Collapse Section	B-13
Multiple Choice	B-14
Dynamic Options	B-14
Triggers and Dynamic Options	B-15
Value Change and Dynamic Options	B-15
Dynamic Control Initialization	B-15
Verify User	B-16
Calculate Net Worth Using If-Else Rule	B-17
Dates and Times	B-17
Duration	B-18
Today's Date and Time	B-18
Date/Time Stamp	B-18
Invalid if Before Today	B-18
Date No More Than 14 Days From Today	B-19
Date No More Than 30 Days Ago	B-19
Update Today's Date to Central Time Zone and Adjust for Daylight Savings	B-19
Hours >= 4 and <= 6 Apart	B-20
Times	B-21
Show Google Maps	B-21
Tenants, Roles, Users	B-21
Item Added to Form	B-22
Item Added - Collapse Other Items	B-23
Tables	B-23
form.load	B-24
form.unload	B-25
Unique ID	B-25
Item Initialization	B-25
Item Added by Init Doc	B-26

Preface

Topics

- Audience
- Documentation Accessibility
- Related Resources
- Conventions

Audience

Using Oracle Process Cloud Service is intended for developers who want to create process applications, administrators who want to monitor process instances and assign user roles, and end users who want to work on application tasks from a browser or the mobile app.

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 that have purchased support 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 Resources

See these Oracle resources:

- Getting Started with Oracle Cloud
- Known Issues for Oracle Process Cloud Service
- What's New for Oracle Process Cloud Service
- Oracle Cloud at <http://cloud.oracle.com>.

Conventions

The following text conventions are used in this document:

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

1

Getting Started with Process

Look here for everything you need to know to start using Process.

Topics

- [What Is Process?](#)
- [Before You Begin with Process](#)
- [How to Begin with Oracle Process Cloud Service Subscriptions](#)
- [About Process Roles and Users](#)
- [Signing In to Oracle Process Cloud Service](#)
- [Quick Tour of the Home Page](#)
- [Quick Tour of the Composer Page](#)

See Oracle Cloud Terminology in *Getting Started with Oracle Cloud* for definitions of terms found in this and other documents in the Oracle Cloud library.

What Is Process?

Use the Process feature to rapidly design, automate, and manage business processes in the cloud.

There are two working environments: the design-time environment, where you develop and test applications, and the runtime environment, where you use and monitor process applications.

The design-time environment:

- Provides business-friendly graphical tools for designing processes, rules, forms, data, and metrics from scratch
- Includes QuickStart Apps for fast and easy rollout of custom business applications
- Provides test environments for refining processes before production activation
- Lets you move process applications (metadata and data) from cloud to on-premises

The runtime environment:

- Makes it easy for you to view, complete, reassign, and delegate tasks
- Lets you stay organized with filters
- Lets you share documents and collaborate with others on your team
- Provides tools to track process flows, view detailed audit trails, troubleshoot, and fix processes

The design-time environment is also called Composer, and the runtime environment is also called Workspace. The terms design time and Composer, and runtime and Workspace are used interchangeably throughout this guide.

Before You Begin with Process

Process has no prerequisites. However, it's helpful if you're familiar with the following technologies, especially if you plan to integrate with applications outside of Process:

- Business Process Model and Notation (BPMN), the standard language for process applications.

www.bpmn.org

Process is based on BPMN, but some of the standards have been simplified for greater usability.

- Web Services Description Language (WSDL), a standard way of describing a web service provider to a web service consumer. You can expose process applications as web services and communicate with web services outside of Process.

www.w3.org/TR/wsdl

docs.oracle.com/javaee/6/tutorial/doc/gijti.html

- Representational State Transfer (REST), an architecture style for designing lightweight, stateless, networked applications that use Hypertext Transfer Protocol (HTTP). Process has a REST API that you can use to integrate with other applications.

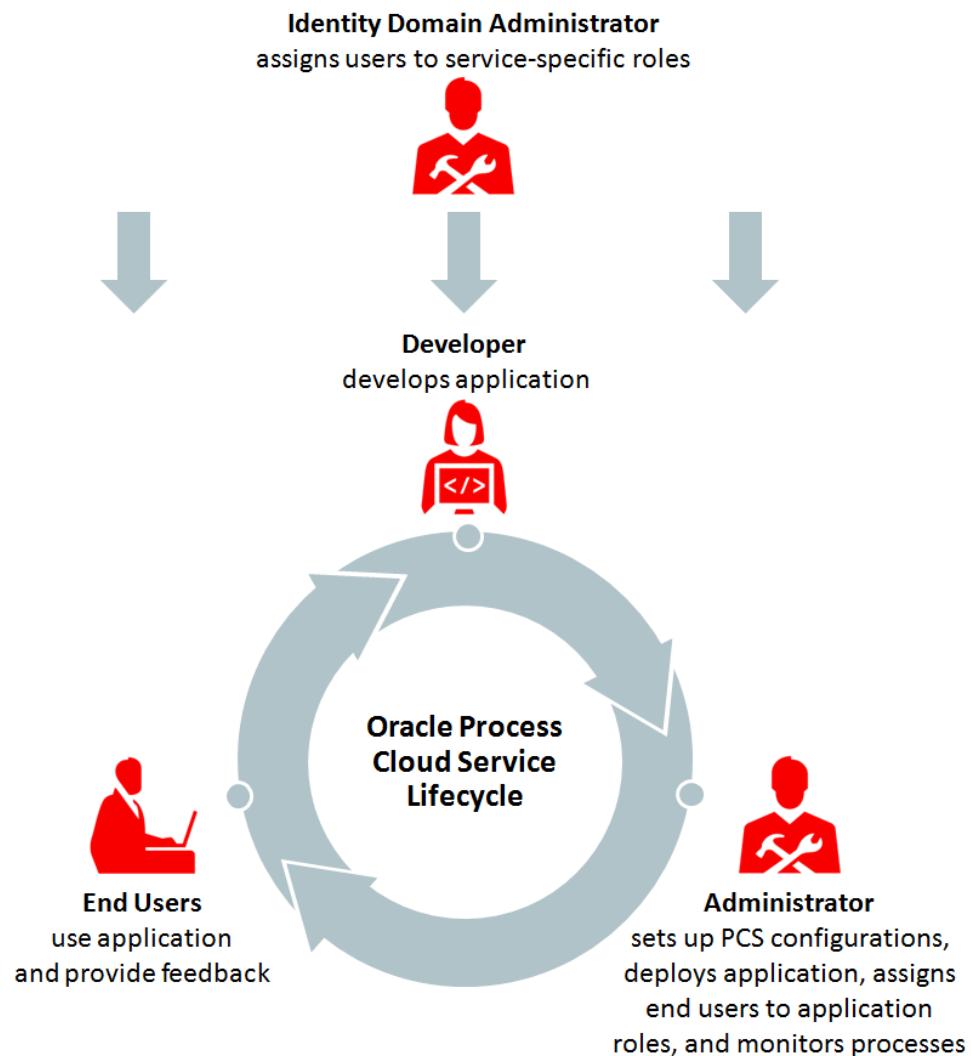
How to Begin with Oracle Process Cloud Service Subscriptions

Here's how to get started with Oracle Process Cloud Service trials and paid subscriptions:

1. Use one of the following methods to get a cloud subscription:
 - Sign up for the free Oracle Cloud credit promotion
 - Subscribe to an Oracle Cloud Service trial
 - Buy a nonmetered subscription to an Oracle Cloud serviceRegardless of which method you choose, be sure to follow the complete setup procedure described in *Getting Started with Oracle Cloud*, and verify that your cloud service is ready to use.
2. Learn about the users and roles.
See [About Oracle Process Cloud Service Roles and Users](#).
3. Create accounts for your users and assign them appropriate privileges and roles.
See [Adding Users and Assigning Roles](#) in *Getting Started with Oracle Cloud*.

About Process Roles and Users

There are various roles to which a user can be assigned to access, administer, and use Process.



Role	Description
Identity Domain Administrator (Administrator)	<p>Can perform all tasks in the My Services application, including user and role management tasks.</p> <p>Oracle Cloud assigns this role to all trial users. See Oracle Cloud User Roles and Privileges in <i>Getting Started with Oracle Cloud</i> for information about this role.</p>
Administrator (<code>service_instance_name.ProcessServiceAdministrator</code>)	<p>Can perform the following tasks:</p> <ul style="list-style-type: none"> • Access all Process components • Monitor and manage processes • Manage user accounts and grant roles to users • Configure connections to other services, such as Oracle Content and Experience Cloud

Role	Description
Developer <i>(service_instance_name.ProcessServiceDeveloper)</i>	<p>Can perform the following tasks:</p> <ul style="list-style-type: none"> • Develop and activate process applications • Share applications with other developers • Start applications • Perform process tasks • Reassign tasks to other users <p>Developers can't access the administration pages in the design-time or runtime environment, and can't create, modify, or delete users for Process.</p>
End User <i>(service_instance_name.ProcessServiceUser)</i>	<p>Can perform the following tasks:</p> <ul style="list-style-type: none"> • Start applications • Perform process tasks • Reassign tasks to other users <p>End users can't access the design-time environment, can't access the administration pages in the runtime environment, and can't create, modify, or delete users for Process.</p>
Oracle Content and Experience Cloud User <i>(service_instance_name.OSN_INTEGRATION_USER_ROLE)</i>	<p>In Oracle Content and Experience Cloud, assign the integrations user to this role. In Process, use this user's credentials to configure the connection to Oracle Content and Experience Cloud.</p>

For organizations with development and production pods, the following additional considerations apply:

- Leave the `ProcessServiceDeveloper` group empty on the production pod.
- Administrators must assign developers to the `ProcessServiceAdministrator` group on the development pod. This ensures developers can test process applications and assign process-specific user roles.
- Administrators must assign at least one person to the `ProcessServiceAdministrator` group on both pods to permit promotion of applications from development to production.

See Adding Users and Assigning Role in *Getting Started with Oracle Cloud* to manage these roles.

In runtime, each application has process-specific user roles. See [Assigning and Managing Roles](#).

In design time, each application has application-specific developer roles. See [About Application Sharing and Collaboration](#).

Signing In to Oracle Process Cloud Service

Before you can sign in to Oracle Process Cloud Service, you must have a user account that defines your credentials (user name, password, and identity domain) and

access rights to the service. In addition, you need the web address (URL) for the service.

When the service is activated, Oracle sends the sign-in credentials and URL to the designated administrator. The administrator then creates an account for each user who needs access to the service. Check your email or contact your administrator for your account credentials and service URL. The service URL looks similar to this one:

`https://hostname:port_number/bpm/workspace`

With your credentials and service URL in hand, you can access Oracle Process Cloud Service through a web browser on a desktop computer or tablet, or use the Oracle Process Mobile app from a mobile device.

When you sign in to Oracle Process Cloud Service from a web browser, you see the following options:

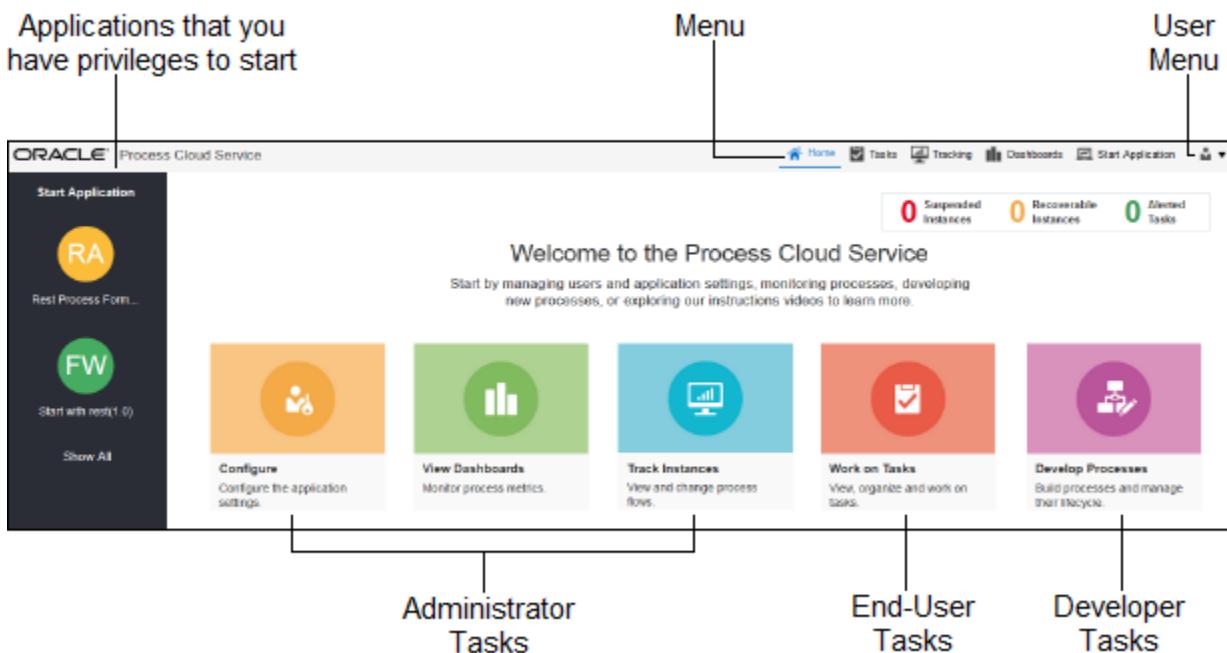
- Start Application
- Work on Tasks
- Develop Processes (for developers)
- Track Instances (for administrators)
- View Dashboards (for administrators)
- Configure (for administrators)

Quick Tour of the Home Page

Depending on your role, use the Tasks runtime environment to work on, monitor, troubleshoot, or administer Process tasks.

When you start Oracle Process Cloud Service, the Home page opens. You only see the functions that you can perform based on your user role (end user, developer, or administrator). For example, the image below shows the Home page for an administrator who has access to all functions.

Use the Home page to manage your work. You can either click the icons or use the application navigation bar at the top of the page.



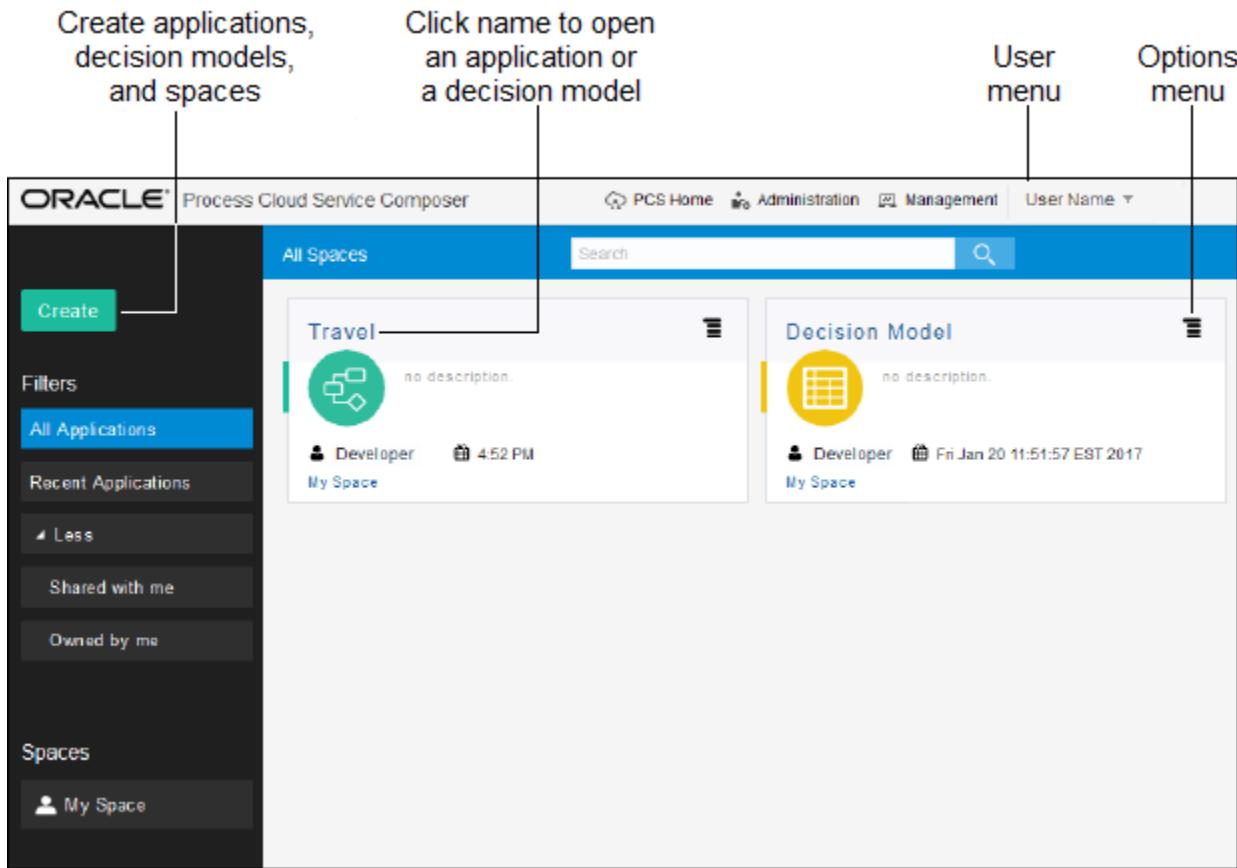
Be sure to take advantage of the resources and videos available in the Learn More section, and also note the location of the user menu. This menu has options for viewing user details, accessing help (including videos and tutorials), setting user preferences, and signing out.

Role	Task
End User	<ul style="list-style-type: none"> Start an application, typically by completing and submitting a form. Work on your tasks. You can work on tasks assigned to you, access detailed information, add information such as comments or documents, and reassign tasks to other participants.
Developer	Create applications , build processes , and manage the application life cycle.
Administrator	<ul style="list-style-type: none"> Configure the application settings, such as connections to other Oracle Cloud services, roles and assignments, runtime credentials and certificates, archive schedules, and notifications. View dashboards to check the status of activated processes and monitor other key process metrics such as workload and cycle time. Track running, inactive, and problematic process instances for the applications available to you. View alerts for suspended, recoverable, and alerted instances. You can go to the instances in these states from each of the alerts.

Quick Tour of the Composer Page

A process application is the core component of the design-time environment and contains all the required resources of the application, including business processes.

When you click **Develop Processes** on the Home page, the Composer page opens. This page provides access to common application-related features.



Use this page to:

- Create applications from scratch, based on a QuickStart App, or by importing
- Create or import decision models
- Create, share, and manage spaces
- Manage your applications, including viewing, unlocking, cloning, downloading, and deleting
- Manage your decision models, including downloading and deleting

You can perform some functions only if your user role allows it. If you create the space, you have the owner role and can perform all the functions listed. If someone has shared a space with you and assigned you the viewer role, for example, you can view the applications inside that space, but you can't make changes.

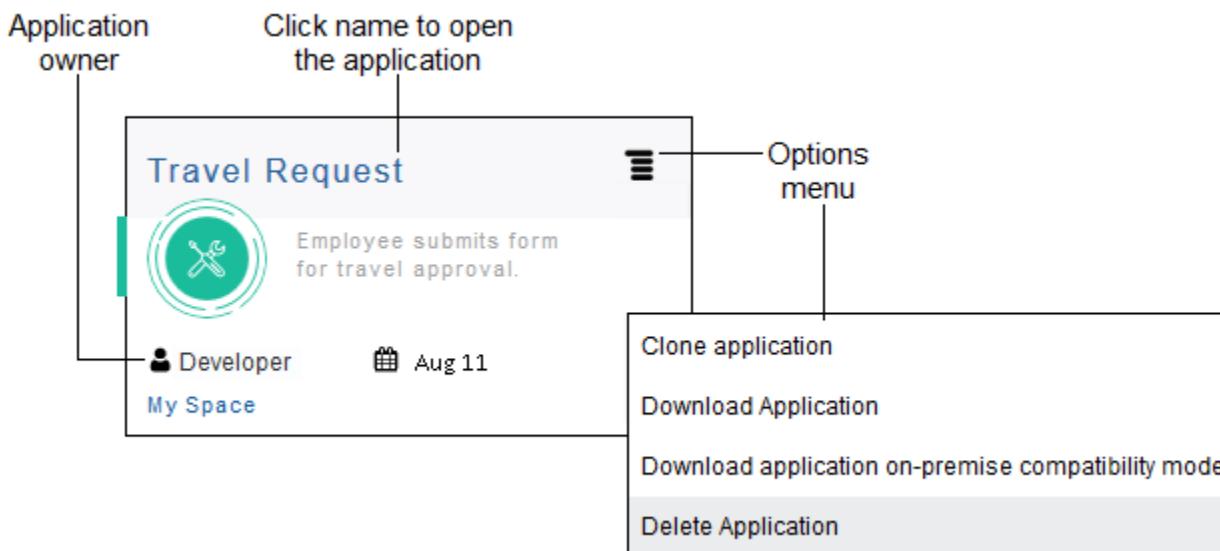
When you sign in for the first time, the system automatically creates a default space for you. The name of this space is based on your browser locale setting. For example, if the language is set to English, the name of the default space is *My Space*. If the language is set to Italian, the name of the space is *Mio Spazio*. Note that if you change your browser locale setting after the default space has been created, the name of the space doesn't change.

Be sure to take advantage of the resources and videos available in the Learn More section, and also note the location of the user menu. This menu has options for viewing your language preferences, accessing help (including videos and tutorials), and signing out.

Now that you have an overview of the design-time Home page, let's take a closer look at some key features that you'll find yourself using most often.

Feature	Description
Create	Provides options that let you: <ul style="list-style-type: none">• Create applications from scratch.• Create decision models.• Create applications based on a QuickStart App.• Import applications that were previously exported and saved as EXP files.• Import decision models that were previously exported and saved as DMN (decision modeling and notation) files.• Create spaces to organize and share your applications. Applications stored in a specific space share the same permissions.
Filters	Displays all applications and decision models in the space or just the ones you have recently worked on. Click More for options that limit the display to applications and decision models shared with you or owned by you.
Spaces	Lets you edit and share spaces you own.

To open any application or decision model, click its name. For example, click an application name to start adding resources and building your processes. Also, you can open the Options menu to clone, download, or delete applications that are owned or shared by you.



Part I

End User Tasks

Topics

- [Working on Tasks](#)
- [General Troubleshooting](#)

Working on Tasks

Process enables organizations to automate virtually any business process. Use it to start an application or work on tasks you've been assigned. Use your browser or the Oracle Process Mobile app to work on tasks. Your work automatically syncs between devices.

In Process, an automated business process is called an **application**, and it contains one or more **tasks** involving humans. Take a travel request process, for example.

- After learning of a travel opportunity, you start the Travel Request application and submit a form.
- The application requires several approvals before you can book travel. Your boss receives an email about the request. To complete the task, your boss clicks the link, signs in to Process, and selects Approve or Reject.
- Your request follows a path through each task in the application until complete. Along the way, you or others may perform additional tasks. For example, you may attach supporting documents justifying the travel or you may view and accept a travel policy document. Or, you and the other task participants might engage in an online conversation about the request.

To use the mobile app, [install and configure](#) Oracle Process Mobile. For browser and mobile access, you'll need the web address to connect to, and your user name and password.

What can I do with Process?

Engage in business processes automated in the cloud.

As a user, you have two main actions:

- **Start Application**, where you activate an application, typically by completing and submitting a form
- **Tasks**, where you view and complete your assigned tasks

The way you and others perform tasks depends entirely on how the application and tasks are set up. But here are things you typically do:

- [Start an application](#) and complete a form
- [Act on assigned tasks](#) such as Approve, Reject, or Request Info
- [Find tasks](#) and view their details and status
- [Upload documents related to a task](#), download them, and [manage them in folders](#)
- [Participate in conversations](#) related to tasks
- [Set out-of-office routing](#) for assigned tasks

If you're an administrator, learn about additional options in [Task Overview for Administering the Runtime Environment](#). If you're a developer, see [Developer Basics](#).

How do I start an application?

Start an application in one of these ways:

- Go to the Home page, and then click **Start Application** in the top menu. From the list of available applications, click the one you want to start.

Alternatively, from the Start Application side pane, click the application you want to start.

 **Note:**

If you see multiple versions of an application, select the **Show Default Versions Only** field. Selecting this field displays the default (primary) version only of each application.

Complete the form, attach documents if necessary, and click **Submit**. This initiates the first task in the application.

Optionally, click:

- **Discard** to discard the changes and cancel this action.
- **Save** to save your changes and complete the form later. The next time you start this application, the saved form appears.

 **Note:**

If the application you want to start doesn't appear on either page, you may not have the correct role. Contact your administrator to assign you the required role.

Completing Tasks

You can act on tasks assigned to you, access detailed information, add documents or attachments, post comments, and reassign tasks to other participants.

[Find the task](#) you want to work on.

[Work on the task](#) by:

- Modifying the task form
- Adding comments or [using conversations](#)
- [Attaching documents](#)
- Taking action on the task, like approving it or [reassigning](#) it to someone else
- Modifying the task priority
- Reviewing the task history



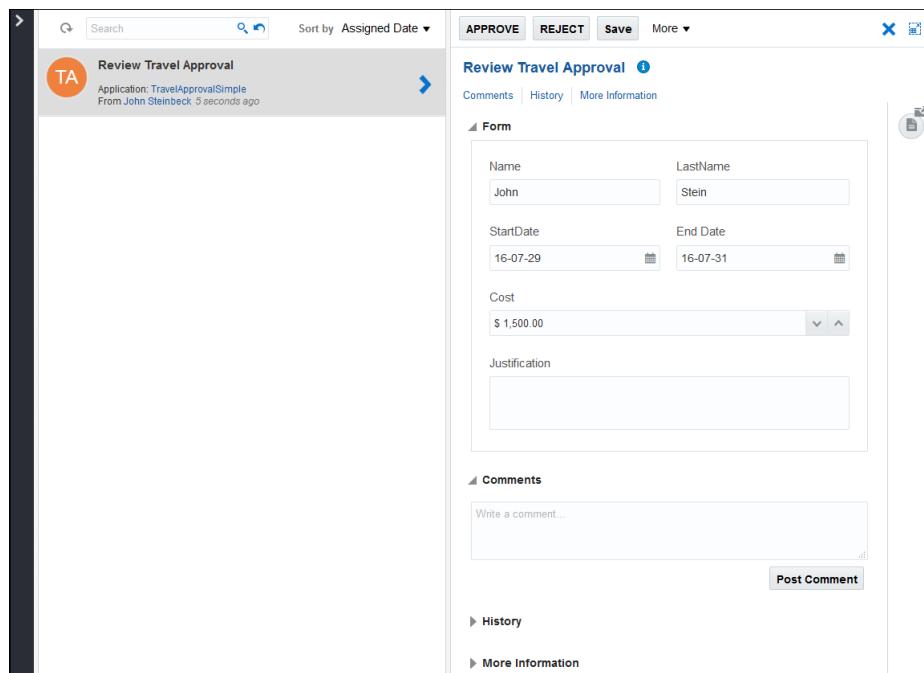
How do I work on tasks?

You can act on tasks, view details about the task, modify the task form, and add information to a task with documents and conversations. When you take action on a task, it moves to the next step in the application.

To work on a task:

1. Go to the Home page, and click **Work on Tasks**.
2. Click **View Details**  in the task you want to work on.

This figure shows the Task Details pane:



3. Optionally, expand the section you want to review or modify:
 - **Form**: View, print, or modify the task form.
 - **Comments**: View or add a comment explaining your decision or information about the task.
 - **Documents**: Depending on how the application was designed, you add attachments in one of two ways. See [How do I add a document to a task?](#)
 - **History**: View the history of the task to understand why it was assigned to you, why it took so long to get to you, and so on.
 - **More Information**: View general information about the task or change the task priority. High priority tasks have a red circle with an exclamation mark.
 - **Conversations**: Collaborate with other task participants to share information, get guidance, make decisions, and so on, if conversations is configured in your environment. See [Can I collaborate with others about a task?](#)
4. Click the action you want to take, or select an action from the **More** drop-down list.

The available actions depend on the application. For example, in a travel request application, the actions may be approve or reject. The **More** actions are:

- **Request Info:** You can request for more information from the task creator or any of the previous assignees.
- **Escalate:** In case you're unable to complete a task, you can escalate it by adding an optional comment in the Comments section and reassign it to your manager (up one level in a hierarchy).
- **Withdraw** (this action is only available to the user that started the application): After starting an application, if you don't want to continue with it, you can withdraw the task.
- **Reassign:** If you're a manager, you can delegate a task to reportees.
- **Claim:** If a task is assigned to multiple users or a group or, you must claim the task. Claim is the only action available in the Task Action list for group or multiuser assignments. Only after a task is claimed, all other applicable actions are listed.
- **Suspend:** You can suspend a non relevant task only if you have been assigned the manager role. A suspension is indefinite. It doesn't expire until **Resume** is used to resume working on the task.

The selected action is performed on the task and it moves to the next step in the application. Unless you're assigned to the next step, the task no longer appears in your task list.

 **Note:**

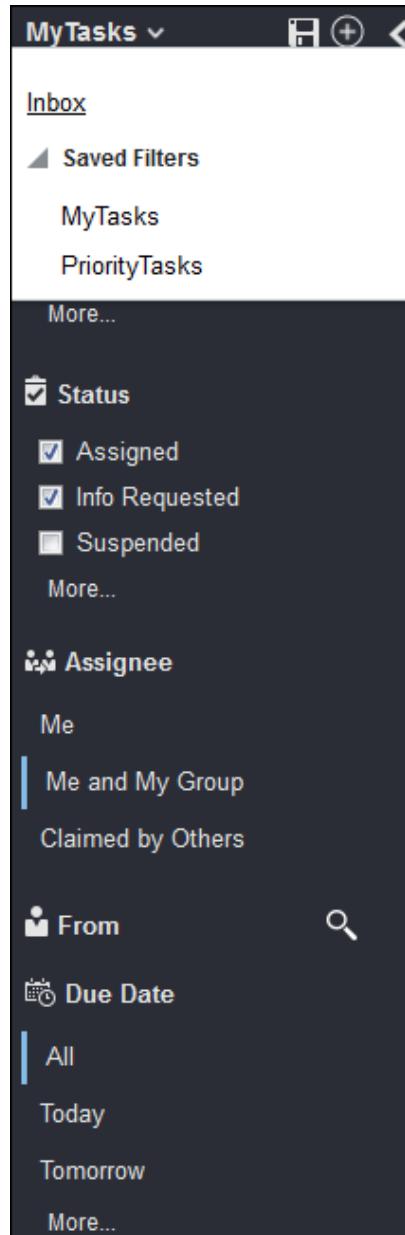
- To perform an action such as Submit or Approve on multiple tasks in the task list, select them by clicking on their icons (for example, ) and choose an action from the top toolbar. The color and initials of the icons vary by application.

How do I find, filter, and sort my tasks?

When you select **Tasks** to view your assigned tasks, you may see many tasks listed. Use the side filters pane to change which tasks are displayed. In the right tasks pane, use the **Search** field to locate a specific task, or the **Sort by** options to change their order. Note that you can click **Show/Hide Filters**  to hide or show the filters pane.

By default, the filters pane's **Inbox** uses standard filter settings, but you can select other filter settings.

- You can filter your assigned tasks by changing filter settings in the filters pane. For example, select **High Priority** to display only tasks assigned to that status. To clear settings and return to the default settings, click **Reset filter**.
- To reuse the filter settings you've selected, save them by clicking **New Filter**  and entering a filter name. The saved filter becomes available in the Inbox.



- To delete a saved filter, click **Inbox**, expand the **Saved Filters** option, click the **Delete** icon, and then click **OK**.
- To search for a task, enter the name of the task in the **Search** field, and then click the **Search** icon. The search results will be displayed.
- To sort your tasks, use the **Sort by** options. For example, select **Title** to sort tasks according to title of the tasks.

Filtering the Task List

You can apply different filters to the task list to view only tasks that match a certain criteria.

Task List Filters

Filter	Description
My Tasks	Displays tasks assigned to you.
High Priority	Displays marked as high priority.
Related to Me	Displays tasks from process instances that you have started and other tasks that you interacted with in the past, for example, a task that you may have reassigned to someone.
Started by Me	Displays tasks from process instances you started.
Administered by Me	For users assigned the Process Owner role, displays tasks from applications they administer.
Reviewed by Me	For users assigned the Process Reviewer role, displays tasks from applications they can review.

Status Filters

You can select one or more status filters.

Filter	Description
Assigned	The assigned state indicates a task has been assigned to a user, role, or group and its ready for action by a user.
Info Requested	Information is requested on a task.
Suspended	Processing has stopped on the task. You can't perform any actions on the task and due dates are suspended while the task remains in this state.
Withdrawn	The task is no longer needed. This is one of the possible ways to close a task.
Error	The task encountered an unrecoverable error. This is a terminal state.
Expired	The task is past its expiration date and has expired. This is a terminal state.
Alerted	When there is a recoverable fault that has happened with task assignment, the task is moved to ALERTED state and assigned to the error assignee.

Assignee Filters

Filter	Description
Me	Displays tasks directly assigned to you.
Me and My Group	Displays tasks that are assigned to anyone, including you, in the groups you belong to.
Claimed by Others	Displays tasks that are assigned to other participants.

From Filters

You can select one or more participants. When you click the Browse icon, a search dialog box opens. You can search users by first name, last name, email, or ID.

Filter	Description
All	Displays all tasks.
[Specific User]	Displays the tasks that were previously processed by this specific user. This option accepts multiple selection.

Due Date Filters

Filter	Description
All	Displays all tasks.
Today	Displays the tasks that are due today.
Tomorrow	Displays the tasks that are due tomorrow.
This Week	Displays the tasks that are due this week.
This Month	Displays the tasks that are due this month.

Application Filters

You can select one or more applications. When you click the Browse icon a list with the available applications appears.

Filter	Description
All	Displays all tasks.
[Specific Application]	Displays the tasks that belong to this specific application. This option accepts multiple selection.

Sorting the Task List

You can sort the task list using different options to have a clear view of the work assigned to you.

Sort By	Description
Due on	Sort the tasks based on their due date.
From User	Sort the tasks based on the user that performed the previous task.
Application	Sort the tasks based on the application to which the tasks belong.
Assigned Date	Sort the tasks based on the date the task was assigned to the current assignee.
Updated Date	Sort the task list based on the date it was last updated.
Priority	Sort the tasks based on their priority.
Title	Sort the tasks based on their title.
Ascending	Sort the tasks in an ascending order. This option is used together with another sorting criteria.
Descending	Sort the tasks in an descending order. This option is used together with another sorting criteria.

How do I add a document to a task?

Some documents are integral to a task—for example, in a home loan process, you would upload a loan application for approval. Some documents just provide

information to other users—for example, in a travel request process, you might need to upload an agenda that shows why you need to take a more expensive flight at a particular time.

To add a document to a task, click **Documents** , and then depending on how the application was designed perform one of the following steps:

- Click **Upload File** and select a file.
- Select the folder you want to place the document in, click **Upload**, and select a file.

This option is available if the application was configured to use Oracle Content and Experience Cloud. Oracle Content and Experience Cloud lets you manage document files and folders in the cloud. See [How can I manage documents in folders?](#)

After a file is associated with a task, you can select the file and select additional actions.

How can I manage documents in folders?

When Oracle Content and Experience Cloud is integrated with Process, you get more robust file features than you do with the file attachment functionality. You can organize files into folders and subfolders, control folder access, and manage documents.

The Documents folder enables you to upload and manage documents and folders in Process. In the Documents folder, you can perform two types of actions:

- Folder actions
- File actions

The Folder actions are common for all files and folders and the File actions only appear when you select a file.

Folder Actions

The following action items are displayed in the toolbar:

- Share this folder (not available for Process).
- Upload files.
- Create a new folder.
- Open the content in Oracle Content and Experience Cloud. To do so, click **Oracle Documents** .

File Actions

Select a file in the Documents folder to perform any of the following actions:

Actions	Description
View	View selected file in Oracle Content and Experience Cloud.
Download	Download selected file.
Move	Move selected items to another location.
Copy	Copy selected items to another location.

Actions	Description
Delete	Delete selected items.
Share	Share selected item.
	This action isn't available for Process.
Upload New Version	Upload a new version of the selected file.
Version History	View all versions associated with the selected file.
Rename	Rename selected item.
Properties	View properties of the selected item.
Sort by	Sort the selected items using the Last Updated, Name Ascending, and Name Descending filters.
	Click More Sort Options  to select the filter.
	View file information either in Grid or List format.
	Click the Views icon, located on the toolbar, to alternate between views.
	Click the Download icon to download files.

Want to learn more about these folder and file actions? See *Managing Your Content in Using Oracle Content and Experience Cloud*.

Can I reassign my tasks to someone else?

You can reassign your tasks to someone else, or delegate them to another user that is helping you with your work.

To reassign or delegate a task:

1. On the Home page, click **Work on Tasks**.
2. On the My Tasks page, select a task, click **More**, and then click **Reassign**.
3. In the Reassign Task dialog box, select **Reassign** or **Delegate**, specify the user or group you're reassigning the task to, and then click **OK**.
 - **Reassign:** The privileges of the new assignee are based on the roles assigned to them.
 - **Delegate:** The privileges of the new assignee are based on your privileges. Use delegate to assign work to a user that is helping you with your work, for example your assistant.

You can use an asterisk (*) as a wildcard when searching for a user or group.

When you reassign a task to multiple users, one of them must claim the task.

Can I collaborate with others about a task?

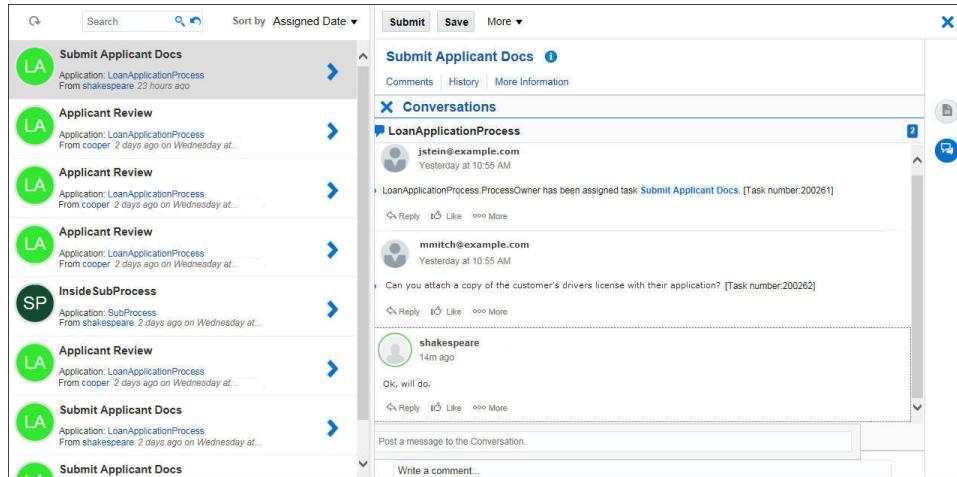
When Oracle Content and Experience Cloud is integrated with Process, you can work with others to share ideas, make decisions, and lend guidance about tasks through online conversations.

To participate in a conversation:

1. On the Home page, click **Work on Tasks**.

- On the Tasks page, select a task, and click **Conversations** .

This action opens the conversation created for the process.



The screenshot shows the Oracle Process Cloud interface. On the left, there is a list of tasks under the heading 'Submit Applicant Docs'. The tasks are listed with their names, application type, and creation time. On the right, a detailed view of a specific task is shown. The task is titled 'Submit Applicant Docs' and is associated with the 'LoanApplicationProcess'. The conversation pane shows messages between 'LoanApplicationProcess' and 'jstein@example.com'. The message from 'jstein@example.com' asks for a copy of the customer's driver's license. The message from 'shakespeare' says 'Ok, will do.' Below the conversation, there is a text input field labeled 'Post a message to the Conversation.' and a placeholder 'Write a comment...'. At the top of the right panel, there are buttons for 'Submit', 'Save', and 'More'.

- Participate in the conversation:

- View current conversation messages
- Post a new message
- Reply to a message
- Like a message
- Mark a message as unread
- Edit a message (available only if current user added the message)
- Delete a message (available only if current user added the message)

Setting Preferences

You can reassign or delegate your tasks when you're out of the office for a period of time. You can also configure the accessibility options to use with Process.

You can configure the following preferences:

- [Out-of-Office](#)
- [Accessibility](#)

How do I set my out-of-office preferences?

You can configure Process to reassign or delegate your work while you're out of the office.

To enable out-of-office:

- Click your user name in the top-right corner, and select **Preferences**.
- Click **Enable Out Of Office**, enter the following details, and click **OK**.
 - Start Date**
 - End Date**

- **Reassign To:** The privileges of the new assignee are based on the roles assigned to them.
- **Delegate To:** The privileges of the new assignee are based on your privileges. Use delegate to assign work to a user that is helping you with your work, for example your assistant.

 **Note:**

A Task gets re-assigned or delegated only when it's assigned to an individual. If a task is assigned to an application role or a group with multiple participants or assignees, the task does not get reassigned or delegated since any one of assignees can approve the task flowing the "Any Single Assignee" approval pattern.

Can I configure accessibility options?

You can configure Process to use a screen reader and to render in high contrast colors.

To configure accessibility options:

1. Click your user name in the top-right corner, and select **Preferences**.
2. Click **Accessibility**, and then select one or more accessibility options:
 - **Use Screen Reader**
 - **Use High Contrast Colors**
3. Click **OK**.

How do I use the mobile app?

First download the Oracle Process Mobile app, and then you can use it to start applications and work on tasks from your mobile device.

Download Oracle Process Mobile on Your iOS or Android Device

1. Download the **Oracle Process Mobile** app to your mobile device, or if you're using iTunes download the app and then sync your mobile device.
2. Start Oracle Process Mobile, accept the license agreement, and review or skip the Welcome pages.
3. Configure the following fields with the connection information that your administrator provided, and then tap **Done**.
 - **Host:** The fully qualified server name where the Process is running.
 - **SSL Enabled:** Selected by default. Tap to turn off if your organization isn't using SSL.

 **Note:**

If SSL Enabled is turned off, the **Port** field becomes active. In the **Port** field, enter the port number where the Process is running.

- **Work Offline:** Off by default. Tap to turn on if you want to work in offline mode.

4. Enter your user name and password to sign in.

Compare Mobile and Browser Features

When it comes to starting an application and working on tasks, you can do many of the same things in the mobile app and the browser. There are a few differences.

Option	Browser	Phone	Tablet
Start an application	✓	✓	✓
Work on multiple tasks at once	✓	✓	✓
Take action on tasks	✓	✓	✓
Filter, search, and sort tasks	✓	✓	✓
Create and save filters		✓	✓
Add attachments	✓	✓	(can add from tablet only, not the cloud)
Add comments	✓	✓	✓
Use conversations	✓		
Reassign tasks	✓	✓ (can use phone contacts)	✓
Track process instances	✓		
View dashboards	✓		
Develop processes	✓		
Work offline		✓	✓

 **Note:**

With Oracle Process Mobile you can also use the device's native features to take photos, browse pictures and photos, and upload them as task attachments.

3

General Troubleshooting

Topics

- [I can't sign in](#)
- [I don't see any applications to start](#)
- [I don't see any assigned tasks](#)

I can't sign in

- Try re-entering the user name and password you were provided.
- Your login information is case-sensitive, so make sure the caps lock key isn't on.
- Make sure you have Internet connectivity.

If you still can't sign in, contact your administrator for details about your account or your password.

I don't see any applications to start

If you don't see application icons in the Start Application pane or page, it's likely you're missing the role needed for access. Contact your administrator to assign you the required role.

I don't see any assigned tasks

This may happen because you aren't assigned the role you need to start applications or work on tasks. Contact your administrator to assign you the required role.

Part II

Developer Tasks

Topics

- Developer Basics
- Creating and Managing Applications
- Developing Structured Processes
- Creating Web Forms
- Creating Basic Forms
- Managing Application Data
- Creating Decisions
- Creating Business Rules
- Integrating Documents and Conversations
- Integrating with Applications and Services
- Troubleshooting Application Development

4

Developer Basics

As a developer, you use Process to create, edit, test, and deploy applications to a production environment. These applications are high-level wrappers that contain all the resources of a business application, including one or more business processes that determine how the application works.

Topics

- [QuickStart Apps, Applications, and QuickStart Masters](#)
- [Creating Your First Application \(a Quick Start\)](#)
- [Personalizing and Promoting QuickStart Apps to the Gallery](#)
- [Learning More about the Components in an Application](#)
- [Managing the Development Life Cycle](#)
- [Setting Preferences for the Design-Time Environment](#)

QuickStart Apps, Applications, and QuickStart Masters

Process is packed with smart tools that streamline designing and building applications for business processes. As the developer, you can focus on creating efficient and effective process flows that make it easier for users to complete their everyday business tasks.

Before you jump right in, review the main features of the types of applications you can create and see what each has to offer.

QuickStart App



- Ready-to-use app
- Pick one from the gallery
- Users without process knowledge can copy, make certain changes, and activate
- Can use as is or opt to customize

Application



- Start from scratch
- Model the process from start to end
- Work in advanced view
- Access to all process features

QuickStart Master



- Convert an application
- Configure what items users can personalize (customization view)
- Change process modeling (advanced view)
- Promote to the gallery for others to use to create QuickStart Apps

There's much more, but this gets you started. It's OK to jump in now. Go ahead. Start exploring and discover everything Process can do.

Creating Your First Application (a Quick Start)

Ready to create your first application? If so, then here's a good place to start. We'll take you through the main steps—from creating the application to making it available to your end users. Along the way, you'll learn new concepts, pick up some orientation and navigation tips, and complete the entire development life cycle for a travel approval application. And you'll still have plenty to discover.

Topics

- [Create from a QuickStart App](#)
- [Customize the QuickStart App](#)
- [Test Activate the QuickStart App](#)
- [Try It in Test Mode](#)
- [Activate the QuickStart App](#)

Create from a QuickStart App

The easiest way to create your first application is to start by creating one from a QuickStart App. A **QuickStart App** is a ready-to-use application with all the implementation details included for you to play, test, and deploy the application.

To get started:

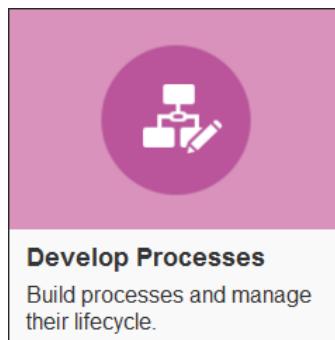
1. Open Composer.

- a. Sign in to your Oracle Process Cloud Service with your user name and password.

The Home page opens. From this page, you can work on your assigned tasks. Depending on your role, the Home page displays other options such as configuring settings or monitoring processes.

You must be assigned the developer role to develop—or also called *model*—a process.

- b. Click **Develop Processes**. The Composer page opens.



Composer is the design-time environment where a developer creates, plays, and tests process applications before an administrator deploys them to actual end users. On the Composer page, the side panel has options related to creating applications, applying filters, and creating spaces.

Spaces are containers for applications. They let you easily share applications with other developers. See [Managing Spaces for Sharing and Collaboration](#).

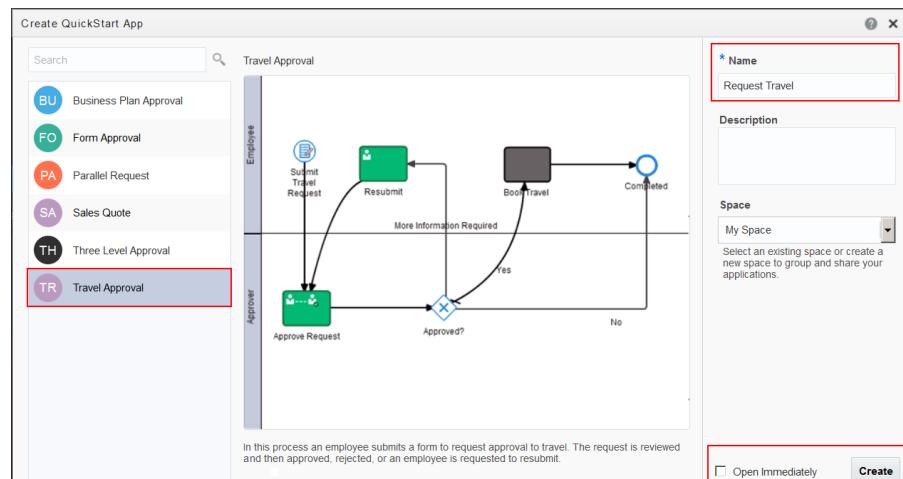
The right panel is your **working canvas**.

2. Create an application.

- Click **Create**, then **QuickStart App**. The Create QuickStart App dialog box lists complete applications that you can use as is or adapt to fit your business needs.
- Take a minute to click each QuickStart App and then view its process diagram and description to get an idea of what's available.
- Select **Travel Approval**.

In Process, an **application** automates one or more business processes to achieve an outcome. This Travel Approval application enables an employee to submit a travel request, which an approver then approves, rejects, or sends back for resubmission.

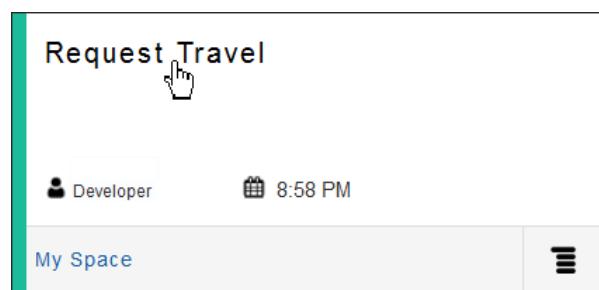
In the **Name** field, enter **Request Travel**. Deselect **Open Immediately** for now (we'll show you another way to open your application) and click **Create**.



After your application is created, notice that it's now listed on the Composer page.

3. Open the application you just created.

- Click the Request Travel application card.



- Take a minute to notice a few things on this page.

The tab identifies your location as **Application Home**.

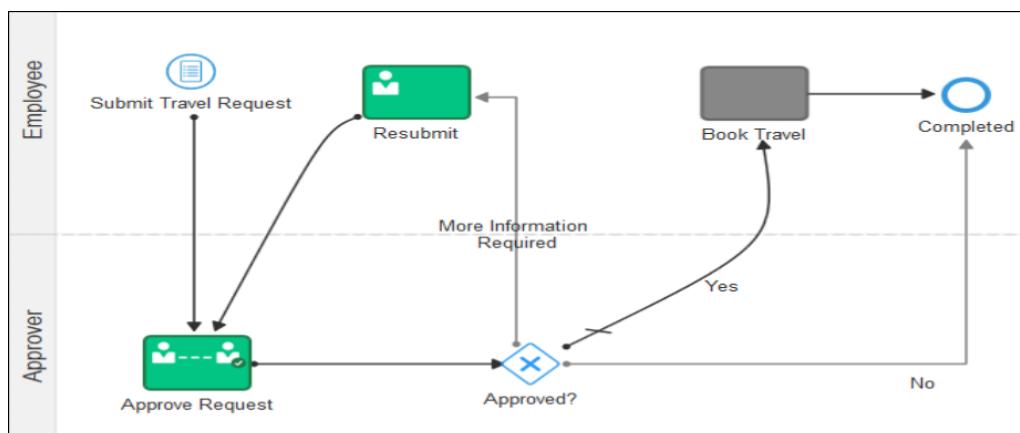
The side panel lists possible components you can use in the application. For example, you can design a **form** that end users complete and submit as part of a process, or create **business types** that store data, such as the form entries.

Because you created the application from a QuickStart App, your application already contains several components. For each component, note that the side panel also shows the number of components currently in the application.

Customize the QuickStart App

In the Request Travel application you just created, let's look at how you can customize the application. Although applications created from QuickStart Apps are ready to deploy as is, you probably want to change and adapt the application for your business or organization.

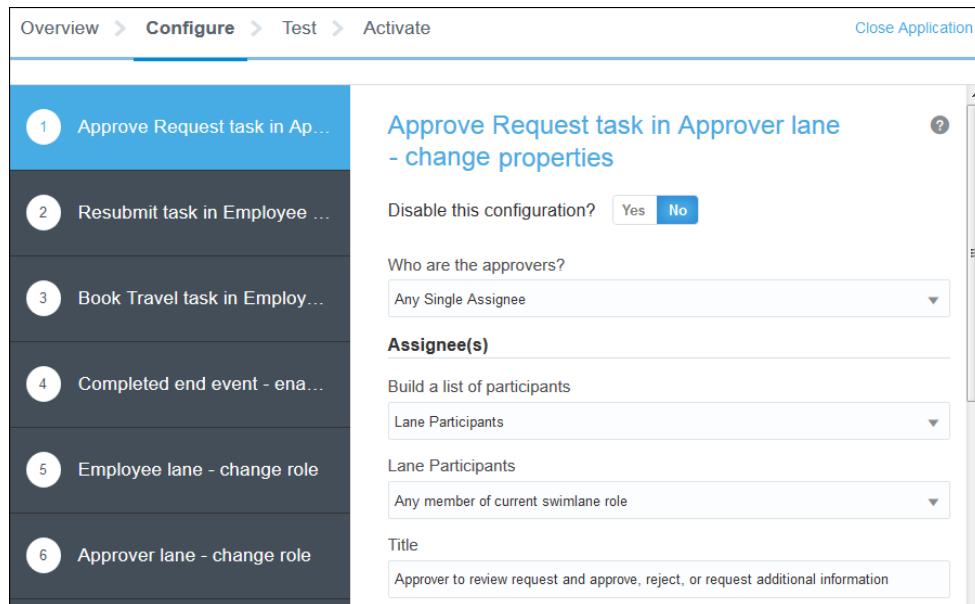
Before you continue, review the process flow for the Request Travel application. An employee completes and submits a travel request form for approval. In the Approve Request task, the designated approver reviews the travel request and takes action. The approver can send the request back to the employee for more information, allow the travel, or deny the travel. Depending on what the approver does, the employee might need to re-submit the form with more information or can book the travel.



To customize the application:

1. Click **Configure**.

Remember that you created this travel application based on an existing QuickStart App. The developer who created the QuickStart Master decides what parts of the application you can customize. For example, in the Request Travel application, you can change settings for the Approve Request task, the Resubmit task, and the Book Travel task.



- 2. Modify settings associated with the approve task.**
 - a. Change who can approve the travel request. For example, in the **Build a list of participants** field, select **Names and Expressions**. Then build the list of approvers by clicking **Add** and searching by user name.
 - b. Scroll the page and change the priority of this task to **High**.
- 3. Modify settings associated with the resubmit task.**
 - a. Click the **Resubmit task in Employee...** link.
 - b. In the Title field, change the text to **Resubmit request: more info needed**.
 - c. In the Due Date field, enter **10d**. The employee will have ten days to resubmit the travel request.
 - d. In the Reminders section, select **Remind Once**, set the interval to **0M5d0h0m**, and select **Before Due Date**. With these settings, the employee will be reminded 5 days before the due date to resubmit the travel form.
- 4. Publish the application.** Publishing creates a version that you can test before it's deployed to end users.
 - a. Click **Publish**.
 - b. Enter a comment about the change you made and click **Publish**.

Test Activate the QuickStart App

In this next step, you'll test your application. **Test activating** activates the application to a runtime environment, which lets you play with and refine it. You can test the various user tasks, flows, and outcomes in the process. You don't activate the application to the production environment until it's ready.

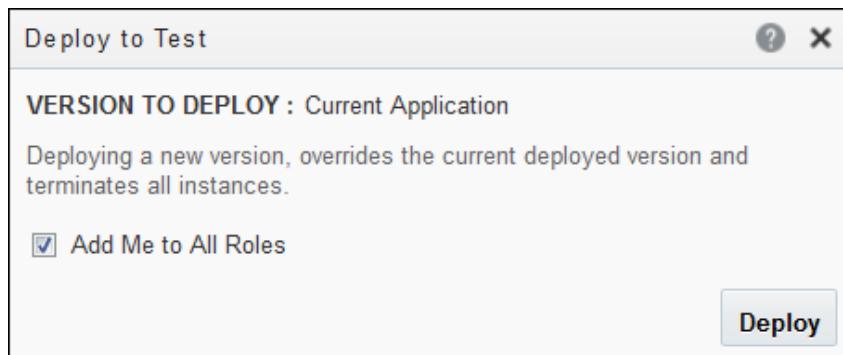
To activate your application to a test runtime environment:

- 1. Click Test.**

The Test page opens. Notice that the drop-down list indicates that the current version of the application is selected for test activation.



2. Click **Deploy**.
3. Make sure the **Add Me to All Roles** check box is selected (so you'll be able to test any task assigned to any user), and click **Deploy**.



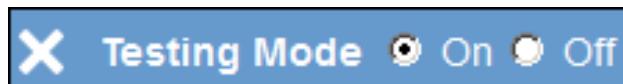
Try It in Test Mode

And now you're ready to try the application in the runtime test mode. Because you added yourself to all roles when test activating the application, you can submit requests in the employee role and also approve or reject requests in the approver role. You're able to test every part of the process.

To test your application:

1. Click **Try in Workspace** to try out your application as an end user. A new browser tab opens displaying the runtime environment, where end users start applications and complete assigned tasks.

Notice the **Testing Mode** option in the toolbar, indicating test activation:



2. Click **Start Application**. Locate your application by its title. Remember, you changed the title to Request Travel. Click its badge.
3. Complete the travel form, which starts the process. The rules that were set up for this form require a value in all fields except **Email** and **Justification**. Click **Submit**.

After you submit, a message confirms that you successfully started the application. Click the badge again and submit another request and repeat until you have submitted three requests.

4. Click **Tasks** in the menu bar. Three tasks that begin with **Approver** display.
 - a. Select the first task to view its details, then click **Approve**. In this case, the process is done and removed from the task list.
 - b. Select the next approval task, and this time, click **Reject**. This task is also removed from the list.
 - c. Select the last remaining approval task, scroll the page, and enter this comment:

Please provide details in your justification and submit again.

Click **Post Comment**, and then click **More Information**.

The screenshot shows a task list on the left and a detailed view on the right. The task list contains three items, each with a pink circular icon labeled 'TA' (Approver) and the text 'Approver to review request and approve...'. The detailed view on the right is for the first task, showing the following fields:

Approver to review request and approve, reject, or request additional information	
Comments History More Information	
Form	
print	
First Name	Last Name
<input checked="" type="checkbox"/> John	<input checked="" type="checkbox"/> Smith
Email	
<input checked="" type="checkbox"/> john.smith@example.com	
Start Date of Travel	End Date of Travel
<input type="text"/> 6/30/2017	<input type="text"/> 7/15/2017
Estimated Cost of Travel	
<input type="text"/> 20000.00	
Justification	
Customer visit	

5. Refresh the task list by clicking **Home**, then **Tasks** from the menu bar. Notice that the task now begins with *Employee* rather than *Approver*. Click to open the employee task, enter a justification, and click **Submit**. The task is removed from the list.
6. Refresh the task list once more and notice the task now begins with *Approver*. Select the task and approve the travel request this time.

You successfully tested each possible scenario in your travel request application.

7. Click **Test Mode Close**  in the Testing Mode bar to exit.
8. Click **Develop Processes** to return to the Composer page.

Activate the QuickStart App

So far, you have created an application from a QuickStart App, viewed and edited the travel approval process, activated the application to a test environment, and then tested each possible task in the process. Your application is good to go. The final step is to make it available to your end users.

Any user who has administrator privileges can activate a QuickStart App to a production environment. As a developer, you may or may not have this privilege.

To activate your QuickStart App to a production environment:

1. Click **Deploy**.
2. Click **Deploy new version**.

The deploy wizard opens. Select which version of your QuickStart App to activate.

Let the wizard guide you through the activation process. Here are a few additional tips:

- You can skip the Customize page. These settings are optional so you can leave all the fields blank.

- On the Deployment Options page, you must enter a revision ID. You might also want to select override and default options.

The revision ID is part of the activation. You can enter any number (including sub-revision numbers such as 1.0.1).

- If you enter the same revision number as an existing instance, then you must also select the **Override** check box. Otherwise, the activation will fail because there's already an instance of the application with that number. Also, selecting **Override** causes all existing instances to go stale.
- If you enter a new revision number, then that version is activated separately alongside any other versions. Your new version will be listed on the Manage Deployed Applications page with the revision number you assigned. Note that if you're activating a new version, then selecting the **Override** check box has no affect.

Optionally, you can specify that this version be marked as the default version. Because applications are activated using a revision number, you can reference an individual version by its revision number. You can also reference the default version. For example, when calling the REST API to initiate a process, you might want to initiate whatever version has been marked as that default instead of initiating a version by its number.

After you successfully activate your application, you—or a user with administrator privileges—must assign end users to the roles defined in the application process. Roles define what end users can do, such as whether they can start the application and what tasks they're assigned. See [Assigning and Managing Roles](#).

Personalizing and Promoting QuickStart Apps to the Gallery

Do you encounter situations at your company where several departments use a business process that is similar, but has slight variations? For example, is the hiring process slightly different for the sales, development, and manufacturing organizations? Or perhaps an order entry process varies from one department to another.

Why not put the power into the hands of department managers (and others) to easily customize the process to fit their business requirements? As a developer, you can take any existing application and convert it to a QuickStart Master. From there, you decide what settings and details in the process users can customize. With a single click, you promote the application to the QuickStart gallery. Users can then create a new application from the QuickStart App and customize the process for their needs.

Topics

- [Convert an Application to a QuickStart Master](#)
- [Configure How the Application Looks in the Gallery](#)
- [Decide What Settings Users Can Customize](#)
- [Promote the Application to the Gallery](#)

Convert an Application to a QuickStart Master

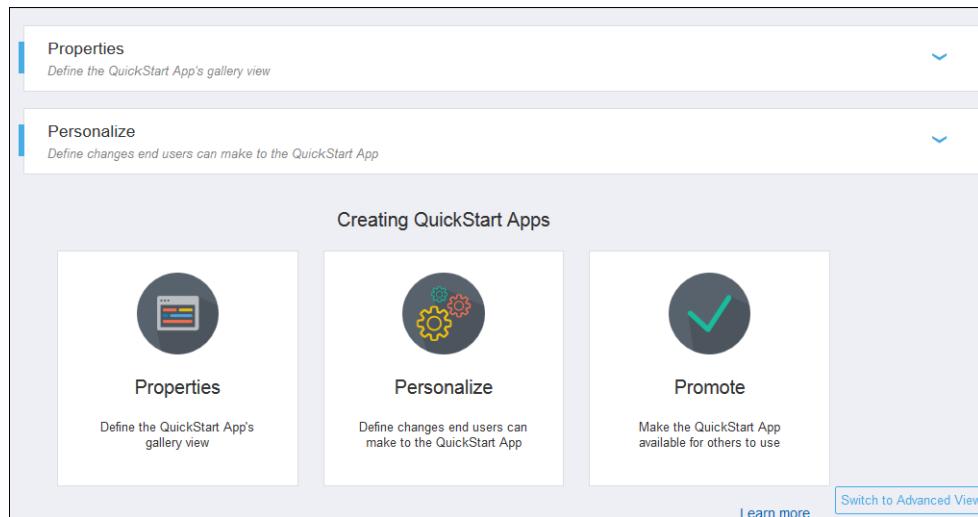
Your first step in promoting a QuickStart App to the gallery is to convert an existing application to a QuickStart Master.

To convert an application to a QuickStart Master:

1. Go to Home page, and click **Develop Processes**.
2. Use the different icons to distinguish QuickStart Apps, Applications, and QuickStart Masters from each other.



3. Open the application you want to convert.
4. Click **Main menu**  and select **Convert to QuickStart App (master)**.

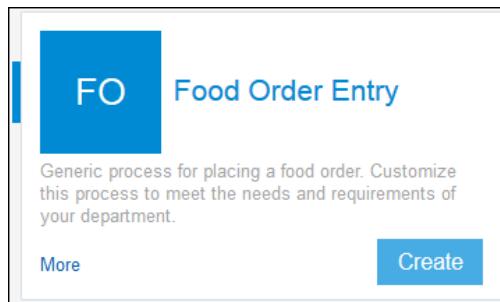


At this point, you're working in what is called **customization view**. From here, you can define how the application looks in the gallery, configure what items users can personalize, and promote the application to the gallery.

Be sure to notice the **Switch to Advanced View** link. If you need to make changes to the process modeling, you can switch to the **advanced view** to access all application features, components, and elements. You can toggle back and forth between the two views at any time.

Configure How the Application Looks in the Gallery

Start by setting the title and description for the application. Users will see this information on the QuickStart App card in the gallery when they go to create their own application.



To set the properties:

1. Click **Properties** to expand the pane.

The Properties pane allows you to define the QuickStart App's gallery view. It includes fields for Name (Food Order Entry), Description (Generic process for placing a food order. Customize this process to meet the needs and requirements of your department.), and Preview Image (a visual representation of the process logic). There is also an 'Update Image' button and a checkbox for 'Allow Advanced View'.

2. Update the name and description. Be sure to enter meaningful and helpful information.

Also, remember that over time your gallery might grow quite large with hundreds of available applications. Users can search by title so that's one more reason to enter a good one.

3. Click **Update Image** to change the image.

You can use any graphic, illustration, or photo. For example, you might want to include a visual representation of the process but use a more interpretative image instead of the traditional flow diagram. Users will see the image when they click the **More** link on the QuickStart App card in the gallery.

4. Decide whether to allow users to switch back and forth between customization view and advanced view. Advanced view gives users access to the Application Home tab.
5. Click **Collapse** to close the Properties pane.

Decide What Settings Users Can Customize

Your next step in creating a QuickStart App is to decide what items and properties in the process you'll allow the user to change.

To configure the settings:

- 1. Click Personalize.**

- 2. Take a minute to review the page.**

- The left pane lists items that are defined in the process flow. The item might be an activity, such as approve, verify, or notify tasks. It might also be for a swimlane in the process.
- The right pane lists the elements and properties that you can configure.

- 3. Select an existing item in the left pane to modify.**

- a. Change the label to be concise and informative. It will appear in the left column. Users will use the label for navigation when customizing the application.
- b. In the Title field, enter a description that gives the user more information about what the task or activity is about. Users will see this title at the top of the page
- c. If it's not clear from the label and title what properties the user can change or why, add instructions in the Help field. When customizing their QuickStart App, users can click **Help**  to view the information.
- d. Click **Choose** to select the properties that the user can change. Each element has unique properties that can be modified. Don't forget to select the preferred value for each property. Users will see this default value when creating their application, but can change it if they want.

For example, properties for an approve task include who can approve the task, how much time they have to approve, what priority is the task, and how often to send reminders about the task deadline.

Which properties can users change?

Done

<input checked="" type="checkbox"/> Who are the approvers?	<input type="checkbox"/> Title	<input type="checkbox"/> Task Summary
<input checked="" type="checkbox"/> Due Date	<input checked="" type="checkbox"/> Priority	<input checked="" type="checkbox"/> Reminders
<input type="checkbox"/> Escalation and Expiration		

4. Add an item to the QuickStart Master that you want to let users customize when they create their own application.
 - a. Click **Add** .
 - b. Skip the Label, Title, and Help Hint fields for the moment. It's easier to enter this information after you know what item you're adding.
 - c. Use the Process, Type, and Item fields to select another item. For example, an activity in an order entry process might be to send an email to customers when they cancel an order. You might want to let users modify parts of the email, such as the To, CC, and Subject fields.

Which element is being customized?

Process	Type	Item
Process	Activity	Notify Customer Cancelled Order in Capture Lane

Allow users to enable/disable the configuration? **Yes** **No**

Which properties can users change?

Done

<input type="checkbox"/> To	<input type="checkbox"/> CC	<input type="checkbox"/> BCC
<input type="checkbox"/> Reply	<input type="checkbox"/> Subject	<input type="checkbox"/> Body

- d. Use the check boxes and drop-down list to select the properties the user will be able to change.
- e. Don't forget to go back to the Label and Title fields and change the default message text.
5. Delete an item that you don't want the user to be able to change.
 - a. Scroll through the items in the left pane.
 - b. Select the item you want to remove.
 - c. Click **Delete**  . Don't worry if you inadvertently delete an item. You can always add it back at any time.
6. Publish your changes.

Promote the Application to the Gallery

At this point, you have converted an application to a QuickStart Master and defined how users can customize the application. Now you just need to promote the application to the gallery. You're promoting a version of your QuickStart App, not the

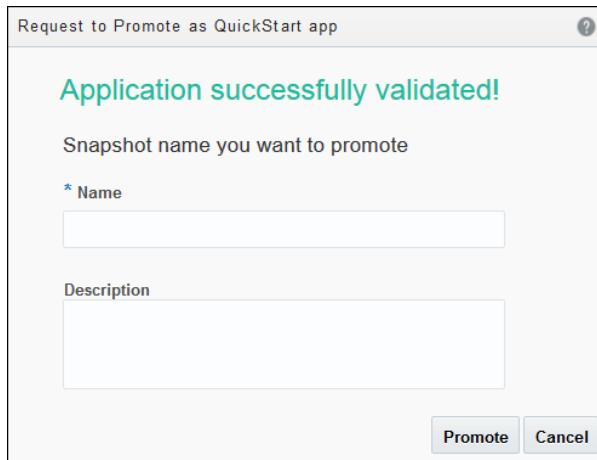
master. You can modify the master at any time and then promote the updated version to the gallery.

To promote the application to the gallery:

1. Test your application.

Even though you converted an existing application, you probably made numerous changes. It's always a good idea to test before promoting an application for others to use.

2. Click **Promote**.



The system validates your application. Be sure to fix any errors before continuing.

3. Enter a name for this snapshot.

This name doesn't appear in the gallery. Instead, the gallery displays the application name you specified in the [Properties pane](#).

4. Click **Promote** again.

Your QuickStart App is now available in the gallery. Go to the Home page, click **Develop Processes**, and then click **Create** and select **QuickStart App**. Scan or search the gallery for your QuickStart App. Any user can use your QuickStart App to configure, test, and activate their own application.

You can [remove a QuickStart App from the gallery](#) at any time. Users will no longer be able to select the QuickStart App to create an application. However, the source application is still available in the list of applications. You can promote it back into the gallery at any time.

Learning More about the Components in an Application

As a developer, it's your job to design and create applications that handle a business process. These applications are made up of components—such as processes, forms, decisions, and documents—that define how the application works.

Processes

A **business process** is a sequence of tasks or activities that result in a well-defined outcome. Business Process Model and Notation (BPMN) elements within the process define the flow and behavior of the application.

 [Video](#)

For many processes, you'll create and edit human tasks. You can configure their duration and deadline, customize the presentation, and add participants and routing. You can also assign users and roles to a human task.

Processes can interact using message start and message end events, send and receive activities, or message throw and message catch events. Processes can also call other processes or include subprocesses.

For developing a process, here are the topics you'll be most interested in:

- [Developing Structured Processes](#)
- [Working with Human Tasks](#)
- [Communicating Between Processes](#)

Forms

Forms define the interface that your application users see during runtime. Think of it as the user interface for a human task or a start form event that starts an application.

You can create forms from the ground up or you can base them on an existing data structure. You can also add controls to a form, and customize the layout of a form.

 [Video](#)

Within a form, you can define the behavior of a form, such as showing or hiding certain form controls based on the state of other form controls.

 [Video](#)

Oracle Process Cloud Service provides two editors: web forms (recommended) and basic forms. For creating and editing forms and their behavior, here are the topics you'll be most interested in:

- [Working in the Web Forms Editor](#)
- [Ready to create a web form?](#)

Business Types

Business types represent real-world concepts or objects, such as a ticket, a request, or an employee. You use business types to create the data structures required in your business application.

Defining how data is stored and manipulated is part of the design and development of an application. You can define complex data types, create data objects, define the expressions used to manipulate the data, define data associations, and define the input and output for your processes.

See [Managing Application Data](#).

Decisions and Business Rules

Decisions are containers for if-then rules and decision tables that use the same input and output data objects. A decision exposes these data objects as a reusable service that multiple business processes can invoke. For example, a decision table can determine whether a manager must approve a travel request based on the travel amount, city, and purpose. See [Creating Decisions](#).

Rules are statements that describe business policies or contribute to key business decisions. You can create and edit decision tables as well as if-then type rules. In your applications, you can assign these rules to the elements in your process flow. See [Creating Business Rules](#).

Integrations

Integrations define how a business process connects to other processes, systems, ICS integrations, REST services, and web services.

You can create connections to ICS integrations, and exposed REST and web services. Your applications can then communicate and exchange data with these services.

See [Integrating with Applications and Services](#).

Documents

You can create folders to organize and store **documents** in Oracle Content and Experience Cloud. These documents can then be used at Process runtime. You can restrict access to only those folders that are relevant to the objectives of the tasks.

You can also define a document or folder that will be used to start a process. For example, home buyers submitting a loan application starts a mortgage process or job applicants submitting a résumé starts the hiring process.

See [Integrating Documents and Conversations](#).

Indicators

Business **indicators** enable you to capture and display metrics specific to your process.

You can select data objects as business indicators to capture business metrics, and then display process metrics in custom dashboards.

See [Working with Business Indicators](#).

Managing the Development Life Cycle

By now, you have created your first sample application, activated it to a test environment, and tried it out in the runtime environment. You also explored the different components, such as processes, forms, and decisions, that make up an application. Before you get too involved with creating more applications, let's look at some key ways you can keep your applications in control and under control.



Managing Spaces and Applications

Spaces group related applications and enable users to collaborate when developing applications. You can create additional spaces at any time, add users who can access the space, and specify what access each user has to the space. For example, you decide whether the user can edit applications in the space or only view them. Spaces help with collaboration as well as organization.

As you create and edit applications, you have options to validate, save, and publish your changes. You can also create snapshots of application changes, view the change history, and import and export applications and snapshots.

See [Creating and Managing Applications](#).

Documenting Your Applications

Providing descriptions, notes, and comments about your applications and processes is a good idea. And it's great advice even if it's coming from a writer.

You can add documentation at the application level, the process level, or the activity level. For example, you can use descriptions to help users go to the correct process, explain what the process does, or point to a process that would better serve their needs. Thoughtful details can provide appropriate context for a report about your applications.

If you're collaborating with other developers, then the team can use the documentation fields to share information such as requirements or reminders. When collaborating with others during the creating or editing process, sticky notes are useful. They're highly visible and easily added and removed.

See [Documenting Applications](#).

Playing, Testing, and Activating Your Applications

As the developer, you can use the application player to test your processes without having to save and activate the application. See [Playing Processes and Testing Applications](#).

Users with owner or editor permissions can activate the application to the runtime test environment so that they can try it out by simulating the end user experience. And users with administrator privileges can activate applications, and perform specific actions such as retire, activate, or shut down applications. See [Managing Active Applications](#).

Setting Preferences for the Design-Time Environment

By setting your preferences, you can configure the language locale to the language used by your browser. If you're an administrator, you can also reset your credentials for accessing management options.

To set your preferences:

1. Click the user name located in the menu bar and select **Preferences**.
2. Select the **Use Browser Locale** check box to use your browser's language.
Note that the language currently set is displayed so that you can determine if changing to your browser's language is required.
3. Click **OK**.

If you have administrator privileges, then the Preferences dialog box includes a **Reset Credentials** check box.

As an administrator, you must enter your credentials (user name and password) whenever you click **Management** in the menu bar. You can skip this step if you select the **Remember Me** check box whenever you sign in.

However, there might be times when you want to undo the Remember Me option. You want to clear your saved user name and password, and be forced to reenter them. If that's the case, then simply select the **Reset Credentials** check box and save the

change. The next time you click **Management**, the Credentials dialog box prompts for your user name and password.

5

Creating and Managing Applications

Let's take a closer look at how you create and manage applications, including sharing spaces to encourage collaboration, testing applications before promoting to production, and generating reports about your applications.

Topics

- [What You Can Do on the Application Home Tab](#)
- [About Application Sharing and Collaboration](#)
- [About QuickStart Apps](#)
- [Creating a New Application](#)
- [Editing Application Properties](#)
- [Validating an Application](#)
- [Saving and Publishing Changes to an Application](#)
- [Playing Processes and Testing Applications](#)
- [Working with Application Snapshots](#)
- [Viewing the Change History of an Application](#)
- [Defining Application Roles](#)
- [Defining Business Parameters](#)
- [Localizing Applications](#)
- [Importing and Exporting Applications and Snapshots](#)
- [Documenting Applications](#)
- [Generating Application Reports](#)

What You Can Do on the Application Home Tab

The Application Home tab provides access to information about a Process application and access to common application-related features.

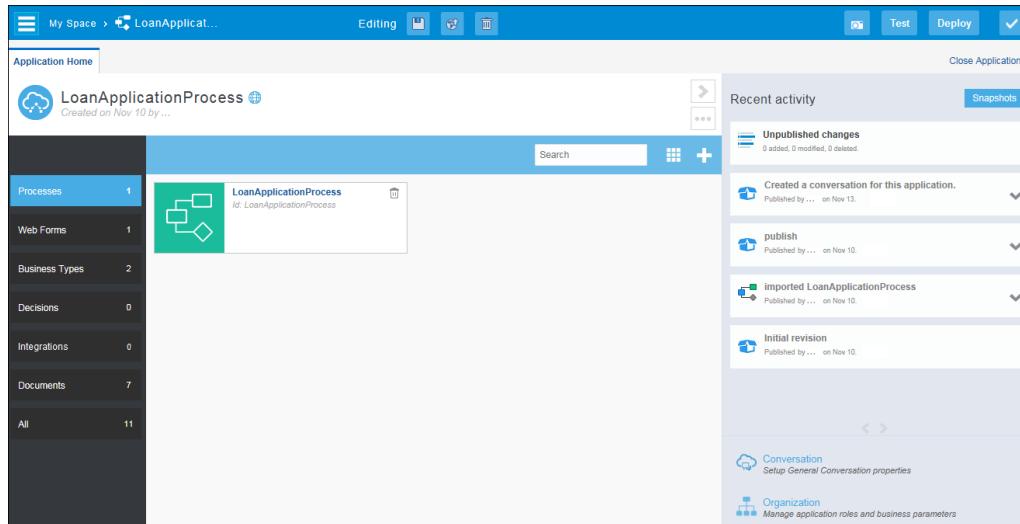
Use the Application Home tab to perform the following types of tasks:

- Edit and view an application's general information.
- Create and manage application components.
- Edit, validate, and publish applications.
- Generate reports.

The Application Home tab is divided into the following sections:

- Application toolbar
- Information pane

- Components pane
- Recent Activity pane
- Conversation link
- Organization link



Application Toolbar

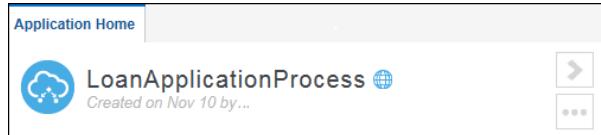
The application toolbar is located across the top of page. It provides access to the main menu as well as to options for saving, publishing, and validating applications.

Toolbar Icon	Name	Description
	Main menu	<p>Opens the main menu for the Application Home tab. From this menu, you can:</p> <ul style="list-style-type: none"> • Import a process model • Export the current or last published application • Convert the application to a QuickStart Master or convert a QuickStart Master back to a standard application • Test the activation of an application and try it in runtime • Activate the application to production • View, add, and delete snapshots • Generate a variety of process reports • Save your changes • Discard your changes since the last publication • Validate an application • Close the current application
	Edit	<p>Toggles to the Edit mode. When you first open a shared application, it opens in View mode. If the application isn't locked by another participant, you can click this icon to change to Edit mode.</p>
	Save	Saves the current application. Also, leaves the application open so you can continue editing.
	Publish	Publishes the application, makes your recent changes available to other participants who belong to this space.

Toolbar Icon	Name	Description
	Discard	Discards changes made to the application since the last publication.
	Snapshot	Lets you create, view, and delete snapshots. Take snapshots if you want to track the changes made to the application over time.
	Test Application	Lets you test your application. You can: <ul style="list-style-type: none"> Use the application player to play your process and try out the flow, including any unpublished changes made to applications. Activate the application to a test environment so you can try it out and simulate the end-user experience.
	Deploy Application	Activates a new version of the application.
	Validate	Validates the application.
	Participants	Displays the number of participants currently watching this application. This icon is only visible for applications that are shared.

Information Pane

The Information pane displays some general details and status information about the application.



By default, the Information pane is collapsed when you open an application.

- To expand the pane, click **More details** .
- To collapse the pane, click **Hide details** .

The Information pane also includes a few tools that let you edit the application's description, add languages, convert the application, and change whether this application uses the documents and conversations features. See [Editing Application Properties](#).

Components Pane

Use the Components pane to view and create the following application components:

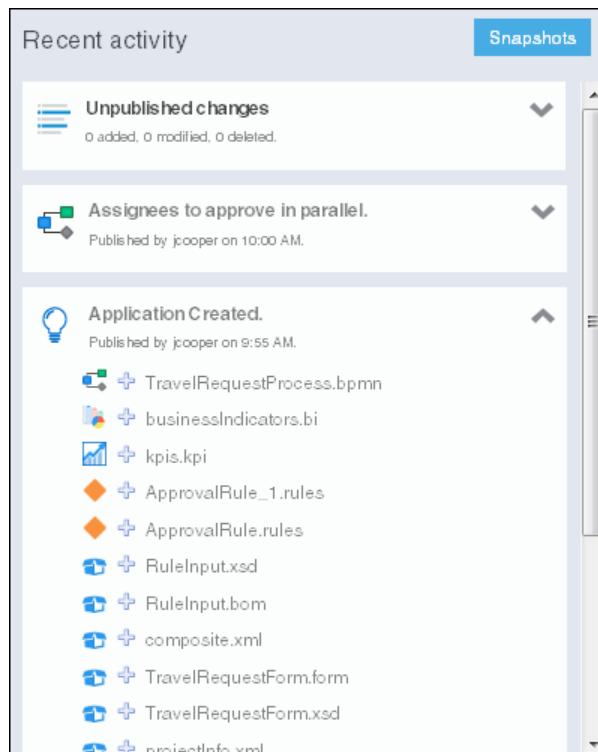
- Processes. See [Developing Structured Processes](#).
- Forms. See [Creating Web Forms](#) or [Creating Basic Forms](#).
- Business Types. See [Managing Application Data](#).
- Decisions. See [Creating Decisions](#)
- Rules. See [Creating Business Rules](#).

- Integrations. See [Integrating with Applications and Services](#).
- Documents. See [Defining Folders and Documents](#).
- Indicators. See [Working with Business Indicators](#).

The number next to each component type indicates how many have been created for the current application. The number next to **All** is the total number of components created for the application.

Recent Activity Pane

The Recent Activity pane provides a history of the changes made to the current application. See [Viewing the Change History of an Application](#).



Snapshots are created from the latest published application. Click **Snapshots** to open the Snapshot dialog box where you can create and view the snapshots for this application. Snapshots let you create backups of your applications and view the changes made to an application over time.

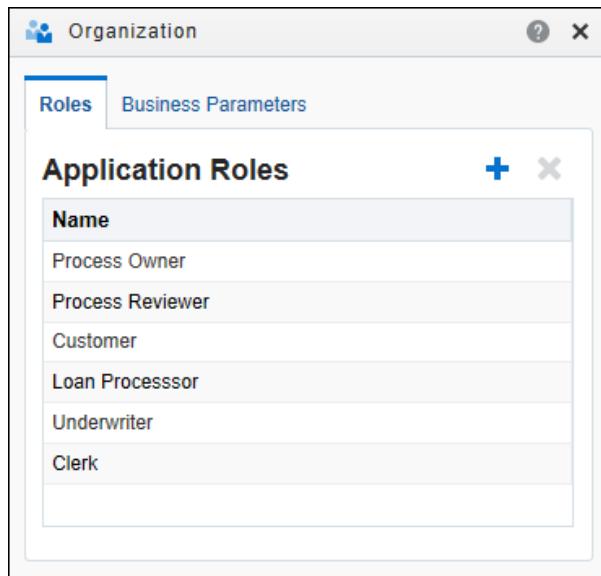
Snapshots provide a record of the changes made to an application during the development life cycle. See [Working with Application Snapshots](#).

Conversation Link

To set up general properties for conversations that are created automatically for each process, click the **Conversation** link. Task participants can share information, make decisions, provide guidance, and so on by participating in conversations when working on their tasks. See [Editing Conversation Properties](#).

Organization Link

To manage the roles and business parameters used in the current application, click the **Organization** link.



- Use application roles to model the users who perform the work your business process represents. Roles define functional categories that correspond to job functions or responsibilities within your organization. You can create and edit the required roles within your process and assign them to swimlanes. See [Defining Application Roles](#).
- Use business parameters to easily create and modify values that you plan to use in your application. For example, you might set the interest rate as a business parameter in a mortgage loan application, and then use and change the rate as needed. See [Defining Business Parameters](#).

About Application Sharing and Collaboration

Process provides features for sharing applications among its users. You can control who has access to view or edit applications.

You share applications with other Process users at the space-level. Applications stored in a specific space share the same permissions. Only owners can change permissions at the application-level. Applications are defined as either private or shared. Only the application owner can view or edit private applications. The application owner and other users who have the correct permissions can edit and view shared applications. See [Creating a New Space for Collaboration](#).

View Mode versus Edit Mode

Shared applications have a View mode and an Edit mode, which determines whether you can make changes or not.

- **View (Read-only):** The application is open for viewing only.
- **Edit:** The application is open for editing.

In Edit mode, you can make changes to the application. When an application is in Edit mode, only the user editing the application can make changes because the editor locks the application. Other users with the correct permissions can view the application, but can't make changes.

The application toolbar shows the current mode.



Application Roles

Application roles define who has access to view and make changes to an application. There are three types of application roles defined as follows:

- **Owner:** When you create an application, you're the owner of the application. You can also define other users as the owner of an application. The owner can perform the following task:
 - Edit the application
 - Activate the application
 - Create an application snapshot
 - Share the application with other users
 - Delete the application
- **Editor:** An editor can make changes to an application.
When a user with the Editor role opens an application, the application is in View mode. If no other users are editing the application, the user can switch to Edit mode and begin editing the application.
- **Viewer:** A viewer can view an application, but can't make any changes to it.

Managing Spaces for Sharing and Collaboration

After creating a space in Process, you share the space with other users. Sharing a space also shares all the applications within the space.

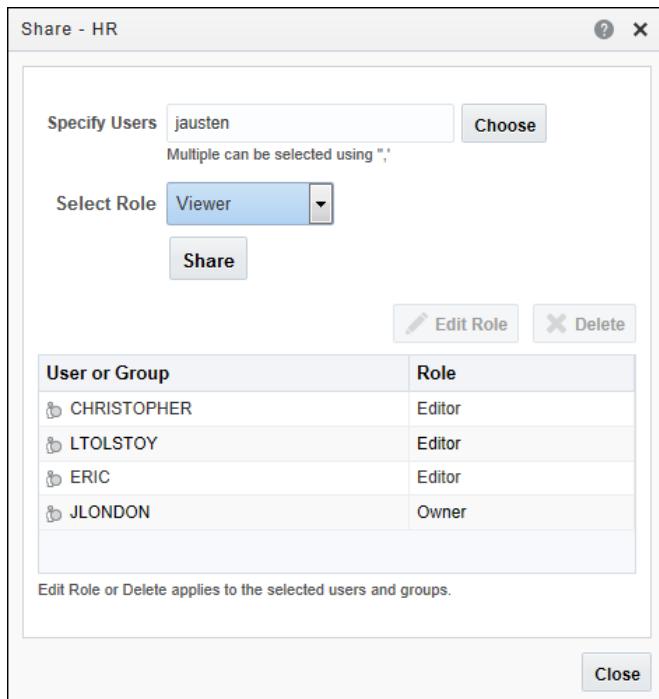
To create a space:

1. Go to the Composer page.
2. Click **Create** and select **New Space**.
3. Click **Create**.
4. Enter a name for the new space and click **Create**.

Your new space is added to the Spaces list.

To share a space with other users:

1. Click the space name in the Spaces list. Note that options expand at the bottom of the list.
2. Click **Share**. The Share dialog box opens.



3. Add users to the space.
 - a. Click **Choose**.
 - b. Use the search options to find the users you want to add to the space. For example, click **Search** to get a list of all users and select the check boxes to add users. You can add one, some, or all users. Just keep in mind that you'll assign the same role to all the users you select.
 - c. Click **OK**.
 - d. Assign a role to the users you selected. The role determines the changes a user can make to an application.
 - e. Click **Share**.

Need to modify a user's role or remove access privileges? Select one or more users in the list, and then click **Edit Role** or **Delete**.

To delete a space and all its applications:

1. Click the space name in the Spaces list.
2. Click **Edit Space** .
3. Click the **Delete space and content** link.

The system prompts for confirmation before deleting a space. Be careful. Deleting a space removes the space and all its applications. You can't recover deleted applications.

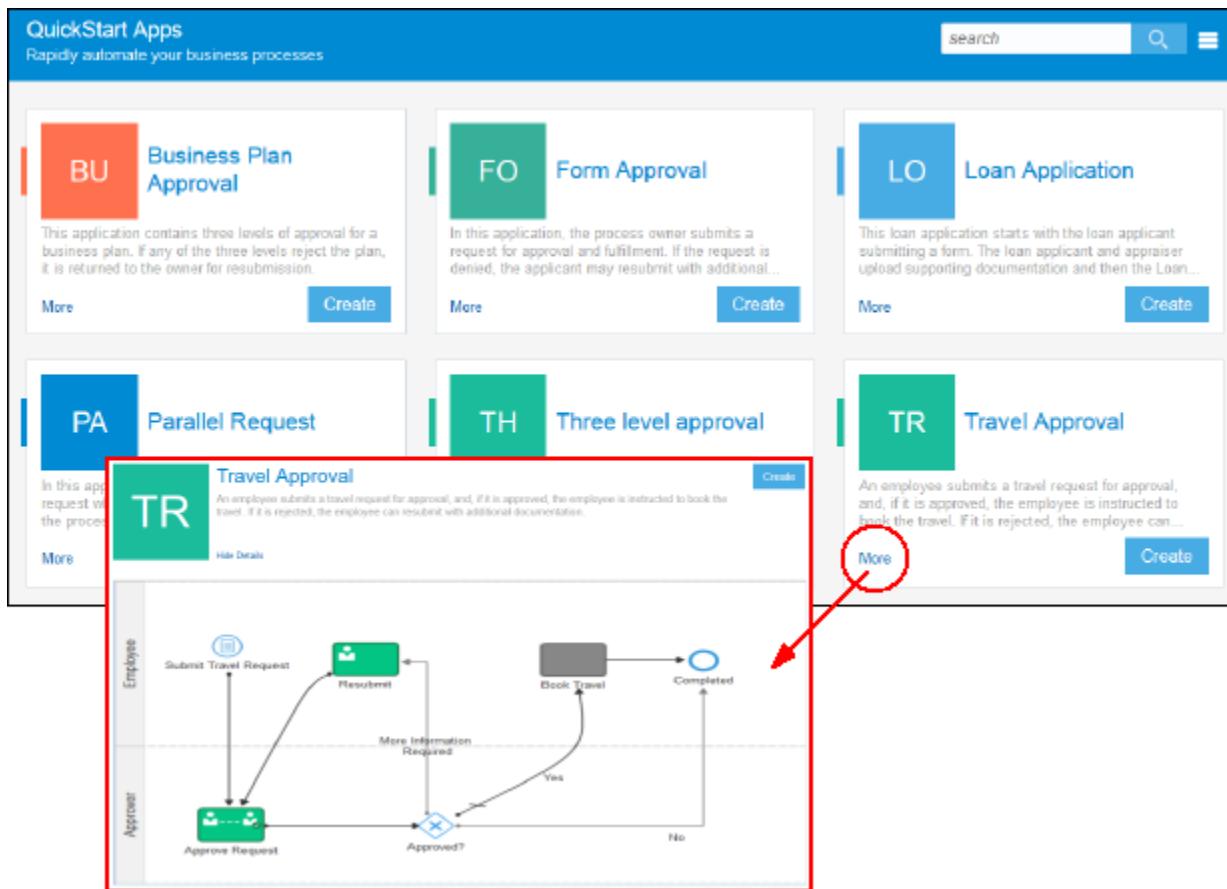
About QuickStart Apps

Select a QuickStart App to quickly create an application and learn about Process. QuickStart Apps include all the implementation details needed to play and deploy

them. Use these applications as a ready-to-use application to deploy as is or use them as a starting point to adapt the application to fit your business needs.

Oracle provides a set of standard QuickStart Apps. For example, the gallery has QuickStart Apps that you can use to create an application for submitting a travel request, processing a loan application, or getting a business plan approved.

In the gallery, scroll the list and read the brief description of each QuickStart App. Click **More** to display a graphical view and description of the predefined process flow.



Developers can create other QuickStart Apps and add them to the gallery. Each QuickStart App will help you rapidly automate your business processes. See [Creating Your First Application \(a Quick Start\)](#).

Creating a New Application

You can create a new application from a QuickStart App or from scratch.

To create an application:

1. Go to the Composer page.
2. Click **Create**, and select one of these options:
 - Click **QuickStart App** to create from a ready-to-use application. Select a QuickStart App from the gallery, click **More** if you want to view the process diagram and additional details, and then click **Create**.

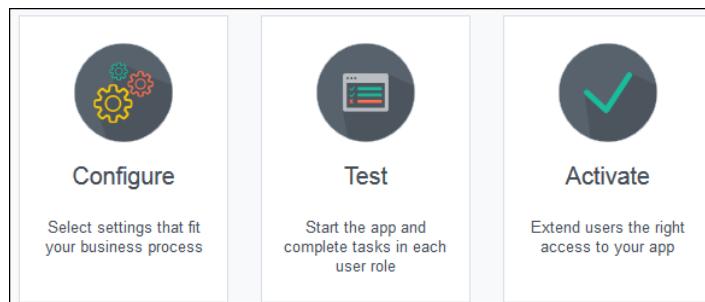
- Click **New Application** to create an application from scratch.
- 3.** Enter information into the Create Application dialog box. Here are a few things to note.

Field	Description
Name and Description	Make sure the application name and description are useful. Give the user a good idea of what the application is all about and why they might want to use it. Good names and descriptions help users distinguish applications with a similar title or purpose. Also, you can't change the name after the application is created. The first letters of the name are used to create a badge for the application.
Space	Select the space where the new application will be stored. All users who are members of the space will have access to the application based on their role. To create a new space, select New Space from the drop-down list.
Open Immediately	Most of the time you'll want to leave the Open Immediately check box selected so you can start configuring and testing your application right away. Deselecting the check box is good if you want to create placeholders for several applications at once and modify them at a later time.

- 4.** Click **Create**.

What happens next and what you do next depends on how you created the application.

- If you created an application from a QuickStart App, the Overview page opens. From here, you can configure the settings to fit your business needs, test each task in the process, and activate your application. And remember, QuickStart Apps are ready to use so you don't need to change any settings. You can skip to test, try out the application, and complete the tasks for each role in the process flow. Adjust any settings if you want, and activate when you're ready.



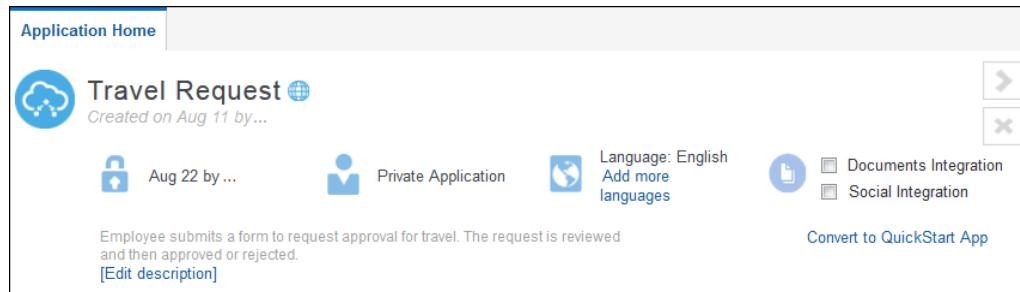
- If you created an application from scratch, the Application Home tab opens. You can dive right in and start to explore. You'll find smart tools that streamline how you configure settings, model the process, and build your application.

Editing Application Properties

The Information pane displays some general details and status information about the application. It also includes a few tools that let you edit the application's description, add languages, convert the application, and change whether this application uses the documents and conversations features.

To edit the properties of an application, go to the Composer page and open an application. By default, the Information pane is collapsed. To expand the pane, click

More details .



The screenshot shows the Application Home page for the "Travel Request" application. It displays the title "Travel Request" with a globe icon, created on Aug 11 by... A lock icon indicates it was created on Aug 22. The application is categorized as "Private Application". The language is set to "English" with a "Add more languages" link. Integration options like "Documents Integration" and "Social Integration" are shown with checkboxes. A note below states: "Employee submits a form to request approval for travel. The request is reviewed and then approved or rejected." with a "[Edit description]" link.

 **Note:**

Your changes are saved automatically. So if you make a mistake, just edit the information again.

Information	Description
Title	Displays the name of the application, when it was created, and who created it. Click Change Locale Name  to the right of the title to change its localized name.
Lock / Unlock	Displays the name of user who is currently editing this application and the date and time the application was locked. If the application is shared and no user is currently editing it, then the application is unlocked. You can click the Edit link to lock the application and make changes to it.
Type	Displays the application type: Shared or Private . If the application is shared, the number of participants is also displayed. Click the link to see details about the users who can access the application and what their role is.
Language	Click Add more languages and use the shuttle controls to add the languages you want to support in this application. Note that this action only adds the selected languages to the current application's set of possible options. After you add languages to the list of options, you can then localize the content. For example, if you want to use additional languages when working with forms, then you must add the languages here first. See Localizing Applications .
Documents and Conversations Options	Available only if your administrator integrated Oracle Content and Experience Cloud with Process. After a connection between the two services is established, you can use documents and conversations in all your process applications. Use these options to configure whether the documents and conversations features are disabled or enabled for a particular application. See Enabling or Disabling Documents and Conversations .

Information	Description
Description	Displays the application description, if the optional text has been provided. Descriptions are one or two sentence expansions of the title to help a user distinguish between applications of similar or same titles. You can add or modify a description at any time.
Convert Option	Includes a link that lets you convert the current application to a QuickStart App, or vice versa.

To collapse the Information pane, click **Hide details** .

Validating an Application

Validating an application enables you to check your application and processes for errors or warnings. Applications that contain errors can't be deployed.

To validate an application:

1. Go to the Composer page and open an application.

2. Click **Validate Application** .

Composer displays a Validation pane for the application and one for each process listing any errors or warnings found.

3. Click **View** to view the errors based on the following options.

View	Options
Scope	<ul style="list-style-type: none"> • Application • Process
Severity	<ul style="list-style-type: none"> • Show All • Errors • Warnings • ToDos
	<p>Note that Errors and Warnings are selected by default.</p>
Flow Object	<ul style="list-style-type: none"> • Show All • Activities • Decisions

4. Click **Export** to export the errors to a Microsoft Office Excel spreadsheet.

Saving and Publishing Changes to an Application

You can save changes to your application as you're editing application components. You must switch to Edit mode to make changes.

To save changes to your application, click **Save**  at any time. All unsaved changes for each application component are saved. You can continue to make changes to the application as necessary. If you're working in a shared application, then the application remains locked and you can continue to edit it.

To make your changes available to other participants, click **Publish** . Published changes are recorded in the Recent Activity pane on the Application Home tab.

To close an application, click **Close Application**.

If you're sharing an application, then a dialog box prompts for how to handle the current lock on the application. To make the application available for other users to edit, be sure to select the **Release Lock** check box.

 **Caution:**

If the application hasn't been published before you save and release, then it asks for whether you want to publish. If you don't publish before you release, your changes are lost.

While editing application components, you can revert your changes and return to the most recently published version of an application.

To do this, click **Discard** , and then click **OK**.

 **Note:**

You can't recover changes that you discard.

Playing Processes and Testing Applications

Use the application player to test the behavior of your business processes at design time. You can test a process using different user IDs without having to activate the Process application.

Topics

- [Mapping Process Roles to Users in Your Organization](#)
- [About Testing Processes Using the Application Player](#)
- [Testing a Business Process](#)
- [About Testing an Application's Activation](#)
- [Testing an Application's Activation](#)
- [Customizing Titles](#)

Mapping Process Roles to Users in Your Organization

Before using the application player, you must map the roles defined in your business process to the users or groups of the organization infrastructure defined in your runtime environment. The application 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 application.
2. Start the player either by selecting **Test Application** from the main menu or by clicking the **Player** ➤ icon.
3. Select the role that you want to map from the drop-down list, which displays all the roles defined in your process.

Role	Identity
PurchaseRequestWithORDS.Process Owner	User PO
PurchaseRequestWithORDS.Process Reviewer	User PR

4. Select the user or group you want to map.
 - a. Click **Choose**.
 - b. Select *User or Groups* 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, select the user or group you want to map.
After selecting a user or group, it appears at the bottom of the selected 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 the player encounters a user task with an unmapped role, it can't continue running the process beyond the human task.

About Testing Processes Using the Application Player

When testing a business process, the application player deploys a version of the application to runtime using a special runtime partition. This allows the player to run a business process using the same environment as a normally activated application.

The application player provides an efficient way of testing the business processes. It uses a runtime environment, accessible from design time, that emulates the real-world behavior of business processes. As the process runs, the player displays a visual representation of the business process showing the path the process instance follows through the process flow. This allows process designers to easily create, test, and revise business processes without having to save and deploy the application and view it.

As a process instance progresses through a process flow, the player displays an animated view of the behavior of your business process. The outline shows the path the process instance takes through the flow elements 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 elements.

 **Note:**

The Instances section on the Application Player tab displays only those instances that are running.

When running the player on a business process, Composer validates the application and deploys the current version of the application to a player partition of the Oracle Process Cloud Service runtime environment. When using the player, you don't have to publish or manually activate the application to view changes while designing a business process.

 **Note:**

Before process modelers can use the application player to test their business processes, an administrator must enable the player. See [Enabling the Application Player](#).

Emulating the Runtime Behavior of Flow Elements

As the player runs through a business process, it emulates the runtime behavior of some of the flow elements in your process.

- **Human Tasks**

When the player reaches a human task in a process, it shows the role or user to select on its behalf. It shows all the possible outcomes as actions. If a form is associated with the task, then the player also gives you the option of launching the form or manually selecting the outcome. If you launch the form, Composer deploys the form and displays it in a separate viewer.

If no form is assigned, the player pauses to allow you to select the role you want to perform the task. It prompts you to select one of the outcomes defined for the human task. **Approve** and **Reject** are defined as default outcomes.

However, the list of possible outcomes depends on how outcomes are defined for the human task. After selecting an outcome, the player continues to the next flow element of your business process.

- **Message Send Events and Send Tasks**

When the player reaches a message send event or a send task event within a business process, it performs these events automatically. It then continues to the instance of the process being called and pauses at the corresponding message catch event or receive task.

In both cases, you must manually return to the parent process. For example, if the send and receive pair is creating an instance on a different business process of the same application, then you must return to the Application Player tab, select the new instance for this process, run the child process, and then return to the parent process.

If the send and receive pair calls an external web service, then you must manually enter the required web service message to continue running the process.

- **Timer Events**

When the player reaches a timer event within a business process, it pauses and waits until you click **Run**. The player then moves to the next flow element in the process flow.

- **Call Activities**

When the player reaches a call activity, it 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 the player reaches an end event, it pauses and displays the **Drill-Up** icon. Clicking this icon causes the player to return to the parent process. If the current process has no parent, the player returns to the Application Player tab and deletes the process instance.

- **Other Flow Elements**

When the player reaches another flow element that causes the instance to wait for some operation or external event, the player pauses. To continue running the process click **Refresh**, which is located at the top of the Application Player tab.

Testing a Business Process

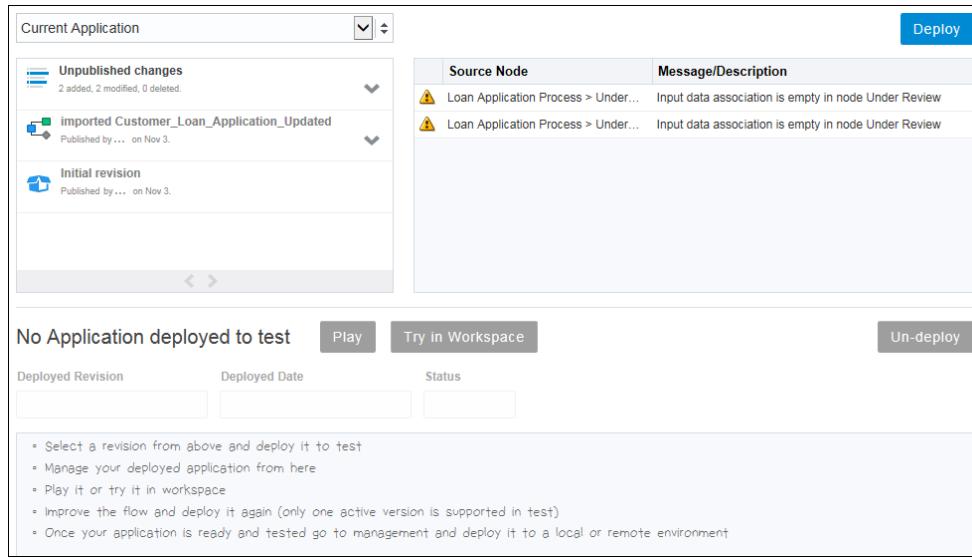
After enabling the application player, you can access the player from the Application Home tab and use it to test the behavior of your business processes. You can access the player from the main menu or the Application toolbar while you're working in Edit mode.

To test the behavior of a business process:

1. Open your process application and access the application player.

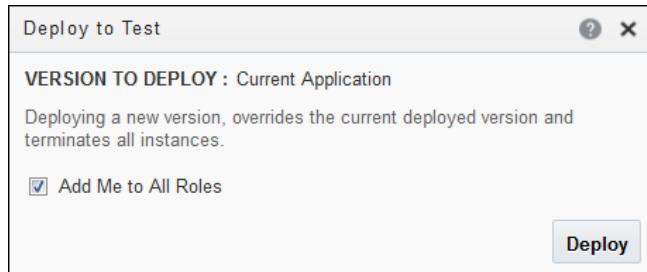
You can access the player by selecting **Test Application** from the menu or by clicking **Test** in the toolbar.

The Test Application tab opens. The application is automatically validated as soon as it's selected.



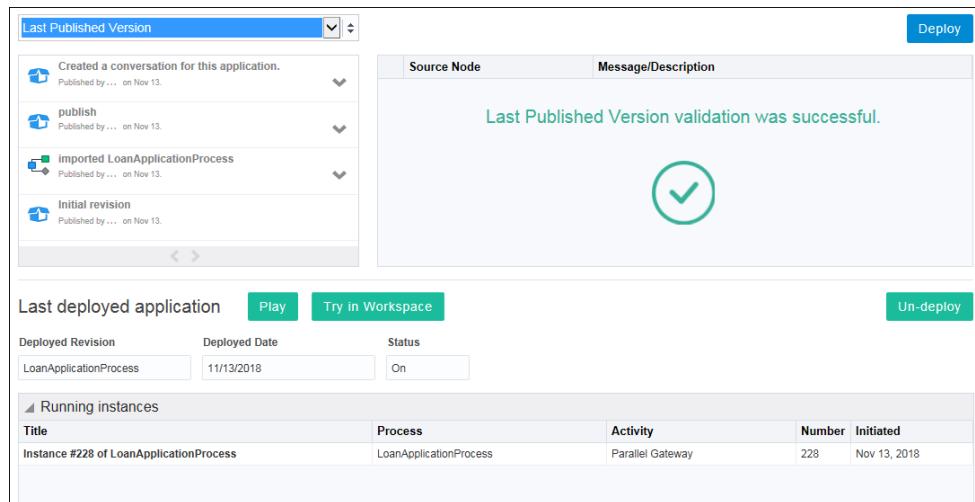
2. Select **Current Application**, **Last Published Version**, or **Snapshot** from the drop-down list.
3. Click **Deploy**.

The Deploy to Test dialog box opens.



4. Select the **Add Me to All Roles** check box so that you can perform the user tasks and click **Deploy**.

A version of the application has now been activated to runtime using a special test partition.



5. Click Play and select the business process that you want to test.

The application player begins running the business process. As it passes through each flow element and sequence flow, it outlines the path it takes through the process flow.

As the player continues running through your process, it stops when the process instance reaches one of the following flow elements:

- Form Start Event
- User Task
- Call Activity
- Service Activity
- Message Event
- Timer Event

You must provide input for these types of elements before the player can continue running through the process flow. To emulate the runtime behavior of these flow elements, see [About Testing Processes Using the Application Player](#).

6. If the player pauses on a form start event or user task:

- a. Click **Play** .
- b. Select the user you want to perform the task.

 **Note:**

If the list of users is empty, then verify that you mapped the process roles to your users correctly. See [Mapping Process Roles to Users in Your Organization](#).

- c. Click **Run** .

Flow Element	Action
Form Start Event	The form associated with this event is launched. Submit the form.
User Task	Select the outcome from the list. The possible outcomes are defined by the human task associated with the current user task.

The player continues to the next flow element in your process flow.

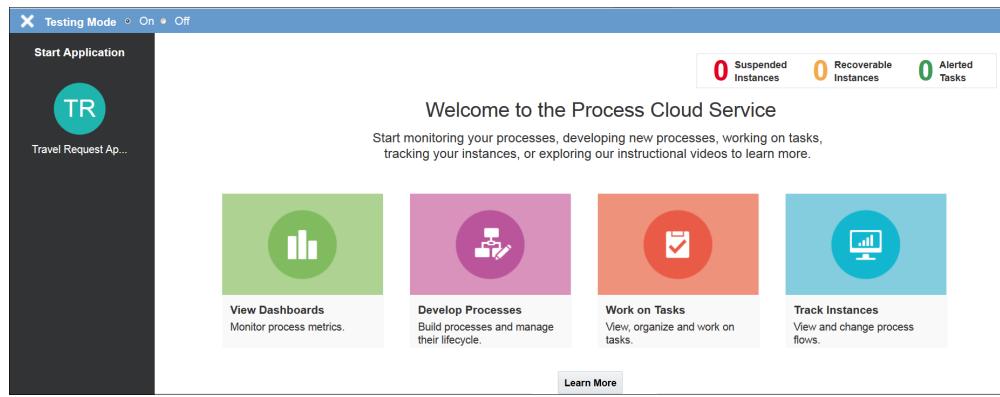
7. If the player pauses on a message catch event or a receive task, it creates an instance of the child process.
 - a. Click **Run**.
 - b. Select the **Player** tab.
 - c. Go to the **Instances** table and select the newly created instance. The design-time environment prompts if you want to close the Application Player tab for the original process. Closing this tab has no effect on the process instances.
 - d. Click **OK**. The player opens the new process instance and begins running the business process from the message start event called from the parent business process.
 - e. Click **Run** for any flow elements that pause the application player as outlined in previous steps.
 - f. When the player reaches the message end event of the child process, click the **Drill-Up** arrow to return to the parent process. The player closes the tab for this child 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, the player continues running through the process from the point where the child process was called.
8. When the player reaches an end event in your process, click the **Drill-Up** icon to end the process instance.

The player returns to the Application Player editor and deletes the process instance.

About Testing an Application's Activation

From the design-time environment, you can activate an application to a test partition in the runtime environment. Note that activation is available only within the same environment where Process is installed.

You must be assigned the `ProcessServiceDeveloper` role to be able to activate an application to the runtime test environment. You can then try out the application and simulate the end user's experience.



Note the **Testing Mode** indicator at the top of the page. When using Testing mode, you can only access the data generated by the applications activated to the test partition. This data is isolated from data in the production environment. You can toggle between Testing mode and Production by selecting either **On** or **Off**.

If you sign out of the runtime environment and want to return to Testing mode, sign in again and then append the following information to the URL:

```
?mode=test&showMode=true
```

 **Note:**

The Dashboard button at the top of the page is inactive. Analytics data isn't published for test activation; therefore, the dashboard doesn't have any test related data.

Testing an Application's Activation

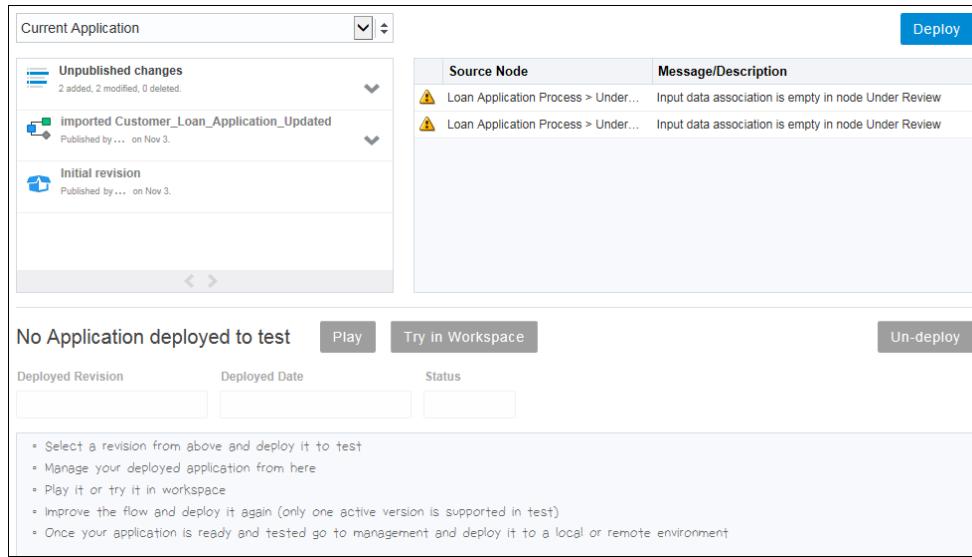
As a developer, you can activate an application to a runtime test environment. You can activate one version of the application at a time.

To perform a test activation:

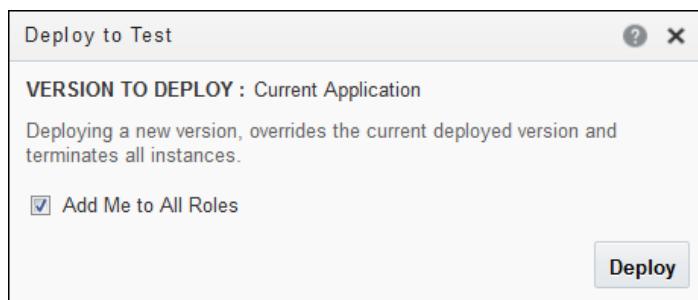
1. Save any changes to your application and publish your application.
2. Click **Test**.

 **Note:**

The application is automatically validated when selected. Validation errors and warnings, if any, are listed on the page. To fix an error, right-click an entry in the list and select **Show in Editor**.

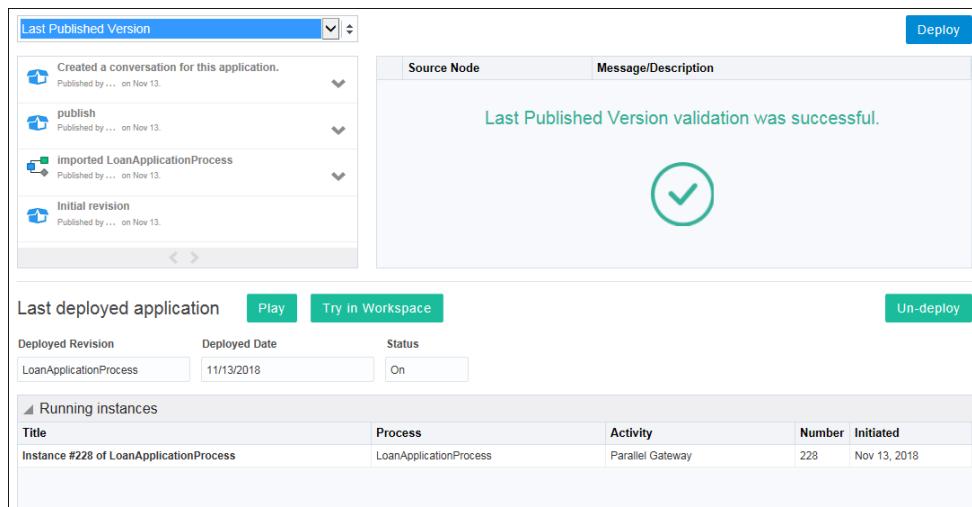


3. Select which version of the application to test. You can test the **Current Application**, the **Last Published Version**, or any available snapshot.
4. Click **Deploy**. The Deploy to Test dialog box opens.

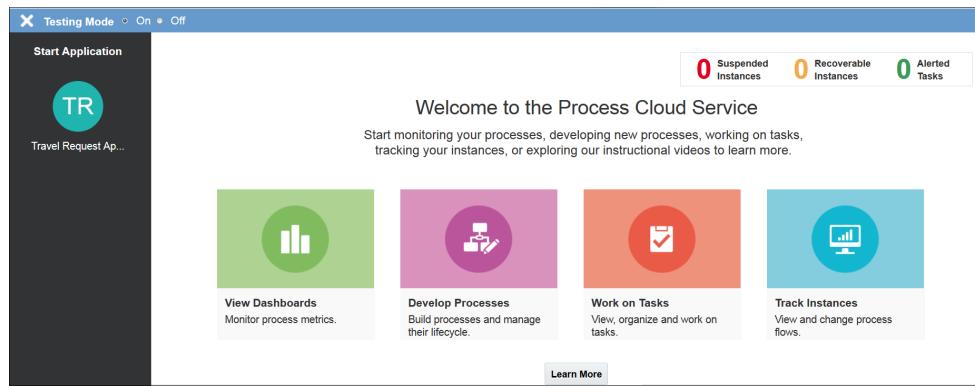


5. Select the **Add Me to All Roles** check box so that you can perform the user tasks.
6. Click **Deploy**.

If the application is activated successfully, then the following page is displayed.



7. Click **Try in Workspace** to go to the runtime test environment and test your process as an end user.



The runtime test environment provides the following information:

- List of applications that you can start
- List of process instances related to you
- List of tasks related to you
- Administration actions: Manage Roles and Manage Credentials.

! Important:

Process Developers can only see roles for the processes for which they're the process owner on the Manage Roles page. All other data is protected and available only to administrators.

Only the keys that are created for test data are shown on the Manage Credentials page. Production keys are protected and only available to administrators.

💡 Tip:

If you sign out of the runtime environment and want to return to **Testing** mode, sign in again and append the following information to the URL:
`?mode=test&showMode=true`

Customizing Titles

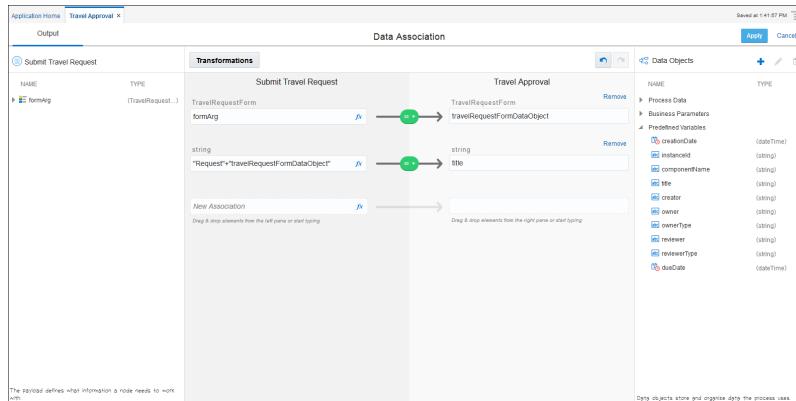
You can customize the name of the process instances and tasks to ensure a better correlation with the business process. The title can be customized during design time in reference to the business process and once the user submits the request, the customized name can be seen in runtime.

Customizing Instance Titles Displayed in Runtime

To customize the title of an instance for display in runtime:

1. Go to the Application Home tab and make sure you're in **Edit** mode if you're editing a shared application.
2. Open the business process where you want to customize.

3. In the process editor, click the **Start Form** activity and then click **Data Association**.
See [Configuring Data Association](#).
4. On the Data Association page, create a **New Association**.
5. Click the **fx** (expression) icon to create a new expression. The expression should evaluate to a string and should contain a static text and your Form data parameters.
See [Working with Expressions](#).
6. In the Data Objects pane, expand **Predefined Variables** and select **title (string)**.



7. Map the **title** data object with the expression you just created, click **Apply**, and then **Deploy** the application.
8. Sign in to Home page, create a new instance, and then go to the **Tracking** page.

The title of the new instance is customized. See [Tracking Process Instances](#).

Customizing Task Titles Displayed in Runtime

To customize the title of a task for display in runtime:

1. Go to the Application Home tab and make sure you're in **Edit** mode if you're editing a shared application.
2. Open the business process where you want to customize.
3. In the process editor, click the **Human Task** activity and then click **Data Association**.
See [Configuring Data Association](#).
4. On the Data Association page, create a **New Association**.
5. Click the **fx** (expression) icon to create a new expression. The expression should evaluate to a string and should contain a static text and your Form data parameters.
See [Working with Expressions](#).
6. In the Data Objects pane, expand **Predefined Variables** and select **title (string)**.
7. Map the **title** data object with the expression you just created, click **Apply**, and then **Deploy** the application.
8. Sign in to the Home page, create a new task, and then go to the **Tasks** page. See [How do I start an application?](#)

The title of the new task is customized. See [Completing Tasks](#).

Alternatively, you can select the **Human Task** activity, click **Open Properties**, and then edit the title using the **Title** field.

Working with Application Snapshots

An application snapshot is a read-only copy of an application at a particular moment. Because snapshots are read-only, you can't open them for editing.

You can:

- Create a snapshot from the last published version of an application
- View the contents of a snapshot
- Export an application based on a snapshot
- Deploy an application based on a snapshot
- Delete a snapshot

Participants with owner or editor permissions can create a snapshot.

On the Recent Activity pane, click **Snapshots** to open the Add Snapshots dialog box. Click **Add Snapshot** , enter a name and description for your new snapshot, and then click **Add**. The snapshot appears in the list of snapshots defined for this application, including the date the snapshot was created and the user ID of the snapshot creator.

By viewing the contents of a snapshot you can compare previous versions of an application with the current one.

Click **Snapshots**, and 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 application.

To return to the active version of an application, click its name at the top of the page.



Users with the Editor role can revert back to a previous snapshot, which basically means setting the time that the snapshot was taken as the current time. Snapshots taken after that time are discarded. To revert to a previous snapshot, click **Snapshots**. Right-click and select **Revert to Snapshot** on the snapshot you want to revert back to.

Users with the Editor role can delete snapshots they created. Users with either the Owner role or the Administrator role can delete any snapshot created by any user.

Caution:

You can't recover a deleted application snapshot.

To delete an application snapshot, click **Snapshots**. Select the snapshot you want to delete and click **Delete**.

Viewing the Change History of an Application

To view the history of major published changes made to an application, use the Recent Activity pane. This pane displays these changes, including creating the application, modifying resources, creating processes, and creating human tasks.

Application changes are displayed in the Recent Activity pane. Click **More Details**  next to a specific change to view the details.

Defining Application Roles

Use application roles to model the users, groups, or system that performs the work your business process represents. Roles define functional categories that correspond to job functions or responsibilities within your organization. You can create and edit the required roles within your process and assign them to swimlanes.

Several application roles are already defined for you:

- Process Owner
- Process Reviewer
- Analytics Viewer
- Automatic Handler—for tasks automatically handled by the system or by any valid user. This role is already defined for you even though it isn't listed on the Roles tab in the Organization dialog box. However, the Automatic Handler option is available when you assign a role to a swimlane in your process.

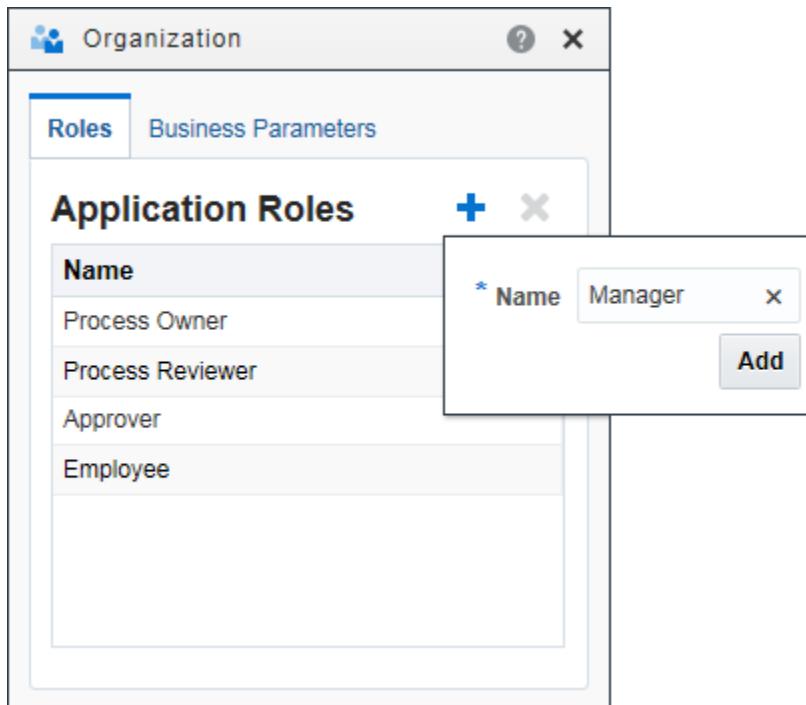
These roles are assigned to the swimlanes in the default pre-defined patterns. You can use these application roles in your processes but they're not required.

You can also [configure roles](#) to create advanced mapping and configuration of application roles, including assigning users to the roles. When an application is activated to runtime, application roles can be mapped to real-world users.

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

To create an application role:

1. Go to the Application Home tab.
2. On the Recent Activity pane, click **Organization**.
3. Select the **Roles** tab.



4. Click **New** , provide a name for the role, and click **Add**.

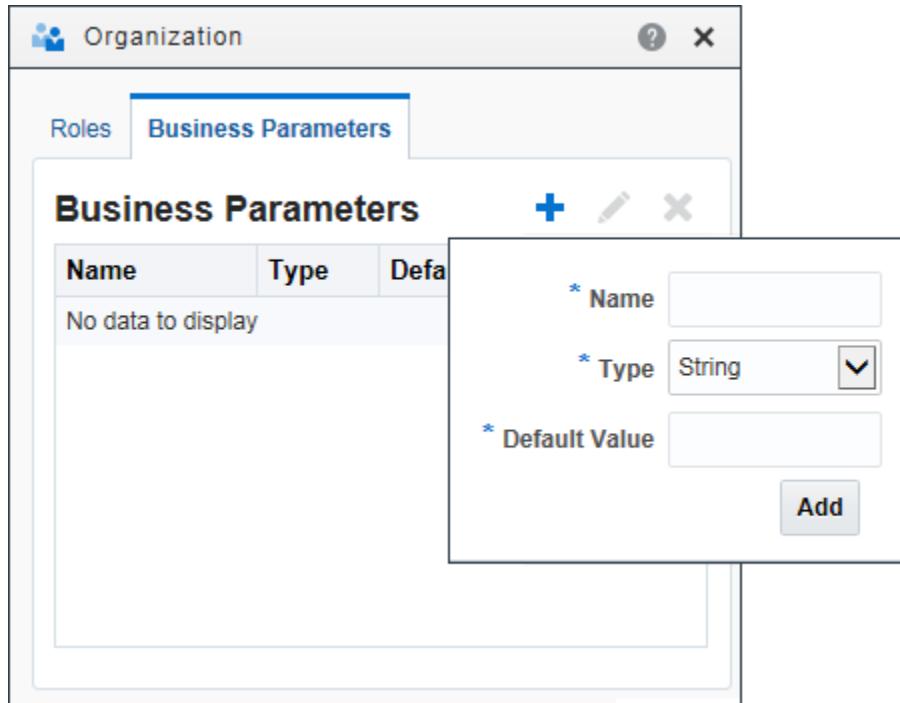
In the Organization dialog box, you can also set a value as a **business parameter** to easily modify and use in an application. For example, you might set the interest rate as a business parameter in a mortgage loan application, and use and change as needed.

Defining Business Parameters

You can set a value as a business parameter to easily modify and use in an application. For example, you might set the interest rate as a business parameter in a mortgage loan application, and use and change as needed.

To create a business parameter:

1. Go to the Application Home tab.
2. On the Recent Activity pane, click **Organization**.
3. Select the **Business Parameters** tab.



4. Click **New +** to create a business parameter.
5. Enter a name for this parameter, the type of parameter (String, Boolean, Double, or Int), and the default value.
The default value is used if no other value is defined.
6. Click **Add**.

Note:

You can edit the name and default value for an existing parameter at any time. You can't, however, edit the type of parameter. If you need to modify the type, delete the business parameter and then create another parameter with the correct type.

In the Organization dialog box, you can also [define application roles](#). Use application roles to model the users or groups who perform the work your business process represents.

Localizing Applications

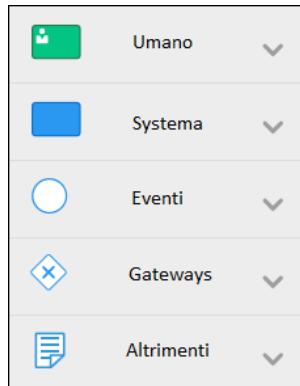
Process enables you to create content such as processes, business rules, forms, and so on, in multiple languages for users. When the application is published and viewed

by end users, the published content displays labels in the localized language based on the users' browser language setting if localized strings are provided.

 **Note:**

It's your responsibility as a developer to localize the content that you create.

When you sign in to Composer, it reads the locale set in your browser, and displays labels, tool tips, and so on in your language, if supported. (If the language isn't supported, then Composer displays all of its labels in English, the default language.) For example, when the browser's locale is set to Italian, the Elements palette in the process editor looks like this:



Applications contain settings that enable you to customize their content (processes, business rules, forms, and so on) by creating localized versions of the content.

To create localized versions of an application's content:

1. Go to the Composer page and create an application.

The new application is displayed on the Application Home tab.

2. Click **More details**  to expand the pane.
3. Click **Add more languages** and use the shuttle controls to add the languages you want to support in this application.

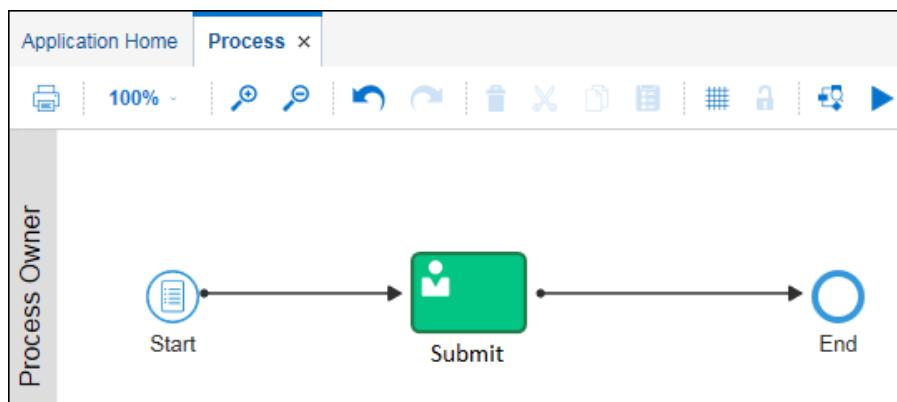
This action only adds the selected languages to the current application's set of possible options.

 **Important:**

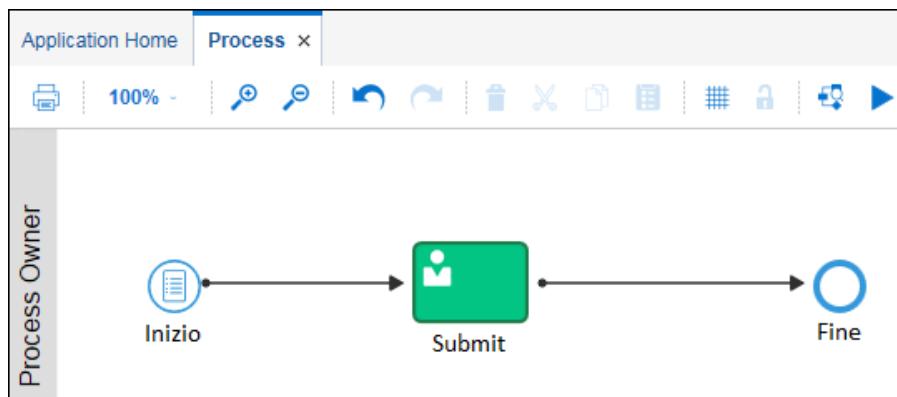
The initial setting for the locale of any newly created application is English. The initial setting doesn't derive from the browser's locale, nor does adding a language to the application's options change the application's initial language setting.

After adding languages to the list of options, you can now localize the content.

4. Create a process, as shown here.



5. On the Application Home tab, select the language you want to use from the language options drop-down list.
- Return to the process editor page. You'll see that the labels, description, and documentation fields still display in the default language of English.
6. Edit and save the labels and descriptions to ensure that the strings are associated with the selected locale.
- For example, after setting the application language to Italian, two process event labels were added and saved, as shown here.



After saving, you can close the process, return to the Application Home tab, and reset the language to English. If you reopen the process, the labels are displayed in English.

7. Toggle between the two languages to view the localized process events in the language you select from the Application Home tab.

Importing and Exporting Applications and Snapshots

You can import and export applications and snapshots as EXP files. (Snapshots are exported as normal applications). With this feature, you can share applications and snapshots with other users directly through the file system.

Topics

- [Importing an Application from your Local File System](#)
- [Exporting an Application](#)
- [Exporting Applications or Snapshots from the Application Home Tab](#)

Importing an Application from your Local File System

You can import an application that was previously exported and saved as an EXP file.

To import an application:

1. On the Composer page, click **Create**, select **Import**, and then select **Import Application**.
2. Click **Browse**, then select the application file (.EXP) you want to import.
3. Enter a name for this application.
4. Select the space where you want to store your imported application. Alternatively, you can create a new space for the application.
5. Click **Import**.

Exporting an Application

Exporting an application to your local file system enables you to share applications.

To export an application to an EXP file:

1. On the Composer page, find the application you want to export.
2. Click **Options** , then select **Download Application**.
3. Select **Save File** and click **OK**.
4. Select a location on your local file system and click **Save**.

Exporting Applications or Snapshots from the Application Home Tab

Applications or application snapshots in Process can be exported to your local file system.

To export an application or snapshot:

1. Open the application or snapshot you want to export.
2. Click **Main menu** , select **Export**, and then select **Last Published Application** or **Export Snapshot**.

The option available is based on whether you open an application or snapshot in step 1. For example, if you open a snapshot, the menu option is **Export Snapshot**.

3. Click **Save File** and then click **OK**.
4. Select a location on your local file system, and then click **Save**.

Your exported application is saved as an EXP file on your local file system.

Documenting Applications

Process allows documentation from the application-level, process-level, and activity-level. None of the documentation fields are required; you can document where appropriate.

Topics

- [Creating Process-Level Documentation](#)
- [Creating Application-Level Documentation](#)
- [Creating Activity-Level Documentation](#)

Creating Process-Level Documentation

Process-level documentation is available as process description, process documentation, process-level documentation links, requirements, and process notes.

Process-level documentation can help the end user make decisions because you provide more detail about the process, links, or requirements.

About Process Description

Like the application description, the process description is a one or two sentence expansion of the title. The description helps make a clear distinction between processes with similar titles.

You can enter a process description at the same time that you create a process or you can add, edit, or delete the description from the General tab of the Properties pane.

Where Used

- Detailed Business Process report
- Business Requirements report
- Process Properties report

In a report that contains many processes, the process description can help the reader quickly go to the correct process. The description can explain not only what the process covers, but also what it doesn't cover along with a pointer to another process.

About Process Documentation

Process documentation is rich text documentation that can be added to a process in the Documentation pane. You can access the Documentation pane by clicking the **Restore Pane**  icon located at the bottom right-hand corner of the process editor page.



If the documentation exists elsewhere you should add links so that the information isn't duplicated. If the documentation is more helpful in the activity documentation or description fields, it should be added to where it's most useful.

Note the **Documentation Type** option, which is located at the top right-hand side of the Documentation toolbar. Use the drop-down list to select if you want your documentation to be created for external (**End User**) or **Internal** use. For your documentation to appear in process reports, you must set this option to **End User**.

When Used

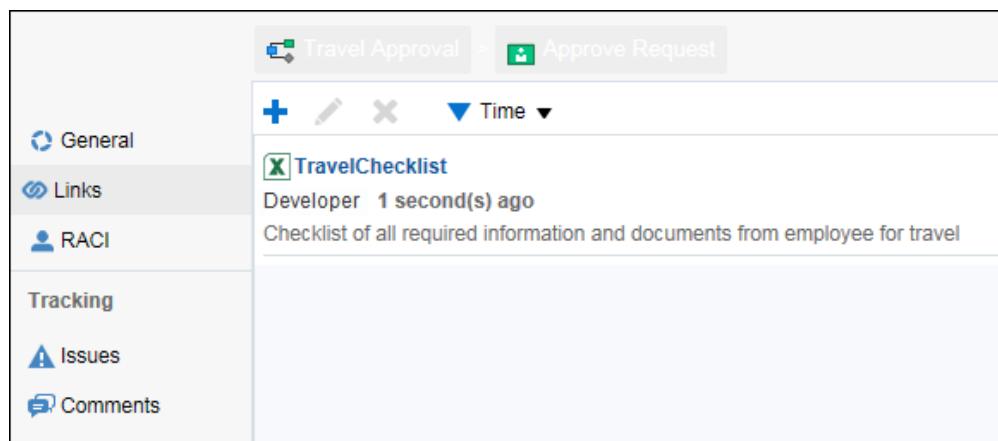
Documentation should be added to a process when the information is important for comprehension or following the process, or when the information doesn't exist elsewhere.

Where Used

- Detailed Business Process report
- Business Requirements report

About Process-Level Documentation Links

Process-level links can be used to link the process to external documentation. However, you must be sure that the link is accessible from Process. If the name of the link is generic or could be confused for another document, use the description to provide more detail. For example, if the sales organization uses several checklists, then the description can help the user determine if the link is to the appropriate checklist. You can enter a process links description in the Links tab within the Properties pane.



When Used

Process-level documentation links can be added to activities, processes, requirements, activity documentation, and process documentation. The description should be short and expand on the title to help users determine whether the link contains the appropriate document.

Where Used

- Detailed Business Process report
- Business Requirements report

About Requirements

Requirements can be added to a process within the Business Properties pane. There are two plain text fields—notes and solution—where more explanation can be entered. You can also add links to existing documentation.

When Used

The requirements feature tracks the status, priority, and difficulty of the requirements of a process. You can add multiple requirements to a process and they can be sorted by various components, such as date, status, and so on.

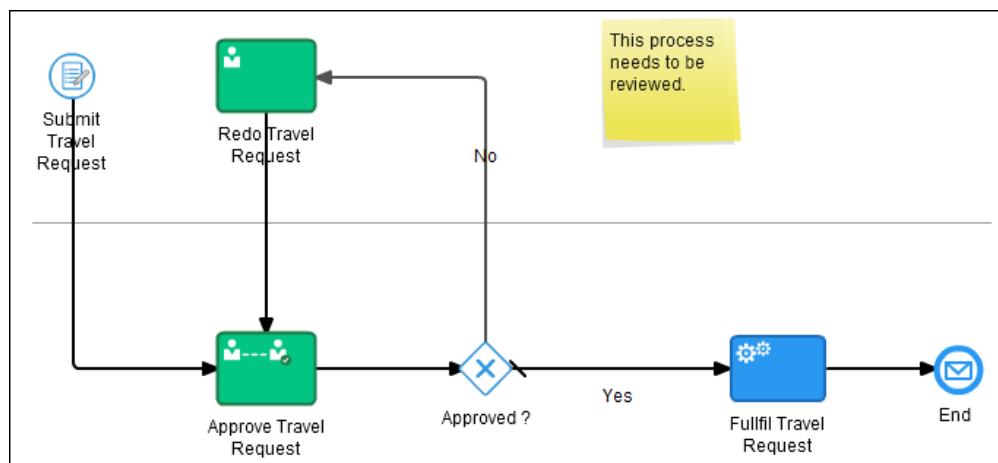
Where Used

- Detailed Business Process report
- Business Requirements report

About Process Notes

A process note is the equivalent of a sticky note—temporary and to be used more as a reminder. They're deleted as soon as the information is used.

To add a process note, drag a Note from the Elements Palette to the process.



When Used

Notes are useful when collaborating with others while creating or editing a process. They're highly visible and can be easily added and removed.

Where Used

Notes don't appear in process reports.

Creating Application-Level Documentation

You can create application-level documentation by adding an application description, which is one or two sentences that help you distinguish between applications of similar titles.

You can enter the application description at the same time that you create an application or you can *add*, *edit*, or *delete* the description within an application.

The screenshot shows the Oracle Application Home interface. At the top left is a blue header bar with the text "Application Home". Below it is a card for a "Travel Request" application. The card includes a blue cloud icon, the title "Travel Request" with a globe icon, and the text "Created on Aug 11 by...". On the right side of the card are two small square icons with arrows (one pointing right, one pointing down). Below the card are several status indicators: a lock icon with "Aug 22 by...", a person icon with "Private Application", a globe icon with "Language: English" and "Add more languages", and a blue circular icon with a document symbol. To the right of these are two checkboxes: "Documents Integration" and "Social Integration". At the bottom of the card is a description: "Employee submits a form to request approval for travel. The request is reviewed and then approved or rejected." followed by a link "[Edit description]". On the far right of the card is a blue link "Convert to QuickStart App".

When Used

When you display all applications available in all spaces, one or more applications of the same title may display. For example, in a multi-national corporation, there might be several *Operations* applications for different countries or divisions. The application description helps you select the appropriate application to open.

Where Used

- Detailed Business Process Report

The application description also provides the appropriate context for a report on an application. Reports are often saved and distributed to viewers who don't have access to Process, therefore, it's important to provide detailed information on the application for which the report was created.

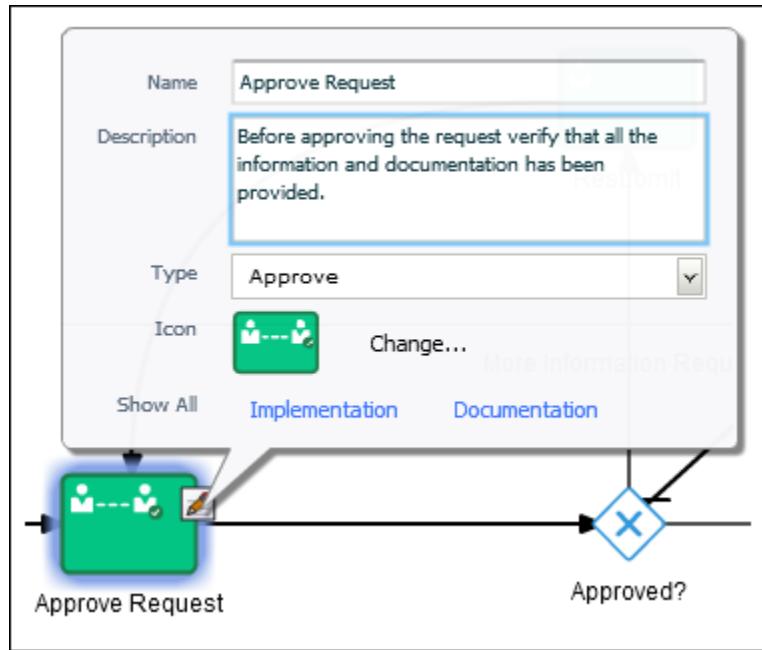
Creating Activity-Level Documentation

Activity-level documentation is available as: activity description, activity-level documentation links, activity documentation, activity comments, activity notes, general issues, and RACI (responsible, accountable, consulted, informed).

You can use activity-level documentation to provide additional information about the activity so that users have a better understanding of the activity, keep records, and determine if the activity is the correct activity when names are similar.

About Activity Description

The activity description should be a brief expansion of the activity name. You can enter the description by editing the activity properties.



When Used

Names of activities tend to be brief. The description is an expansion of the name to help with the understanding.

Where Used

- Detailed Business Process report
- Business Requirements report

About Activity-Level Documentation Links

Links can be added to activities, processes, requirements, activity documentation, and process documentation. The description should be short and expand on the title to help users determine whether the link contains the appropriate document. Activity links are located in the Links tab within the Business Properties pane.

When Used

Customers often have a rich store of existing documentation that can be referenced from the process. You can add multiple links to an activity, and these links are active in the process reports.

Where Used

- Detailed Business Process report
- Business Requirements report

About Activity Documentation

Activity documentation is rich text documentation that can be added to an activity in the Documentation pane. If the documentation exists elsewhere, you should add links so that information isn't duplicated. If the documentation is more explanatory elsewhere, for example, in the description field or as a comment, it should be added there instead.

Note the **Documentation Type** option, which is located at the top right-hand corner of the toolbar. Use the drop-down list to select if you want your documentation to be

created for external (*End User*) or *Internal* use. For your documentation to appear in process reports, you must set this option to **End User**.

When Used

Documentation should be added to the activity when the information is important for comprehension or the information doesn't exist elsewhere.

Where Used

- Detailed Business Process report
- Business Requirements report

About Activity Comments

Activity comments can be added in the Comments tab within the Business Properties pane. Once added, a comment can't be edited or deleted.

When Used

Activity comments are like notes in that they're brief, but unlike notes, they're permanently attached to the process and appear in several process reports. You should use comments when it's important to keep a record.

Where Used

- Detailed Business Process report
- Business Requirements report
- Issues and Comments report

About Activity Notes

An activity note is the equivalent of a sticky note—temporary and to be used more as a reminder. You can delete them as soon as the information is used. Drag a note from the Elements Palette and drop it on or near the relevant activity.

When Used

Activity notes are useful while collaborating with others when creating or editing a process. They're highly visible and can be easily added and deleted.

Where Used

Activity notes don't appear in any process reports.

About General

The General tab tracks the cost, what application systems the activity interacts with, and time to complete the activity.

When Used

General information is used for process improvement and discovery; it can be important to track the cost of performing the activity as well as how long it takes to complete.

Where Used

- Detailed Business Process report
- Business Requirements report

About Activity Issues

The activity issues should be used to track specific issues including their severity, priority, and resolution status. They can be added in the Issues tab within the Business Properties pane.

When Used

Issues can be used during process improvement, testing, and discovery; it can be important to track the issues that are discovered. Issues can be sorted by severity, priority, resolution status, and date.

Where Used

- Detailed Business Process report
- Business Requirements report
- Issues and Comments report

About RACI

The RACI feature tracks who is responsible, accountable, consulted, and informed regarding an activity. You can add RACI information in the RACI tab within the Business Properties pane. To access, click the relevant activity, click the **Actions** icon, and select **Open Properties**, and then select **Business Properties**.

When Used

The RACI data can be used during process improvement, testing, and discovery to ensure the proper roles are involved in the activity.

Where Used

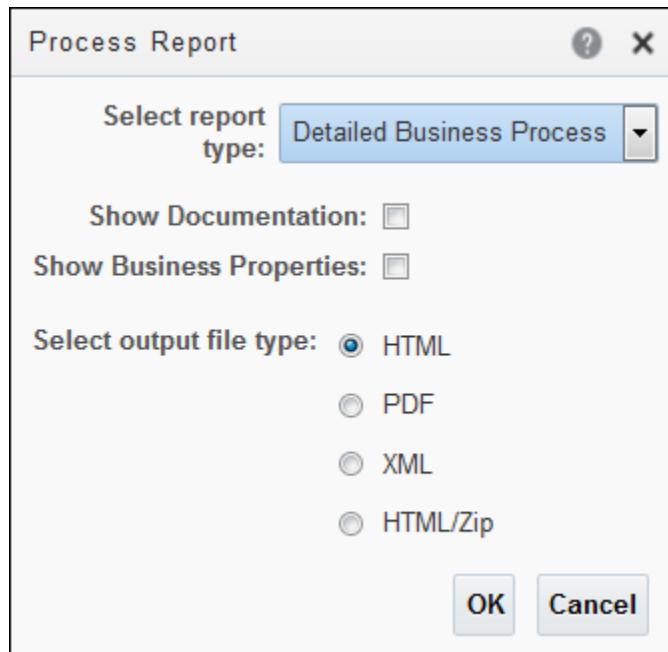
- RACI report

Generating Application Reports

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

To generate a process report:

1. Go to the Composer page.
2. Open the application that you want to get reports for.
3. Click Main menu  and select **Process Report**.



4. Select the report you want to generate. Also, select any optional settings.
 - To include descriptions and other documentation in the report, select the **Show Documentation** check box. The type of documentation that appears on a specific report depends on the report. See [Documenting Applications](#).
 - To include information such as requirements, links to attachments, and tracking of issues and comments, select the **Show Business Properties** check box. Business properties can be at the application-level or the process-level.
 - Be sure to select your preferred file format for the report. The default is HTML.
5. Click **OK**.

6

Developing Structured Processes

A key part of your application is the business process. When developing a structured process, your first step is to determine the people and roles required to complete each task that requires user interaction. You then use the various elements, such as human tasks, system activities, and other events, to design the flow of your process.

Topics

- [About Business Processes](#)
- [Creating a Business Process](#)
- [Working with Process Roles and Swimlanes](#)
- [Working with Elements](#)
- [Communicating Between Processes](#)

About Business Processes

A business process is a sequence of tasks that, after 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.

Although applications are higher level wrappers that contain all the resources of a business application, the processes within the application determine how the business application works. Within a business process, Business Process Model and Notation (BPMN) flow elements define the flow and behavior.

Note:

BPMN is an industry standard notation for defining business processes. Process supports BPMN 2.0.

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

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

Process Type	Description
Synchronous Service	Is invoked from another process or service synchronously. In a synchronous service, the calling process waits until the process completes before continuing.
Asynchronous Service	Is invoked from another process or service asynchronously. In an asynchronous service, the calling process doesn't wait until the process completes before continuing.

Process Type	Description
Manual Process	Require user interaction. Manual processes can begin with a form, a message, a document, or an empty (none) start event.
Reusable Process (Reusable Subprocess)	Can be called by a BPMN process. In BPMN terminology, reusable processes are often called <i>Reusable subprocesses</i> . Use the call activity to call reusable subprocesses within your business process.

Process Instances

A process instance refers to a specific instance of a business process. For example, the Loan Process business process 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 loan application is submitted and approved, and defines the types of people responsible for performing that work. In contrast, a process instance refers to a specific loan application and the specific people responsible for approving it.

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

Process Tokens

Process token is an abstract concept in BPMN. It refers to the current point of execution within a business 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.

Flow Elements

Flow elements are the BPMN components that represent the work performed within a business process.

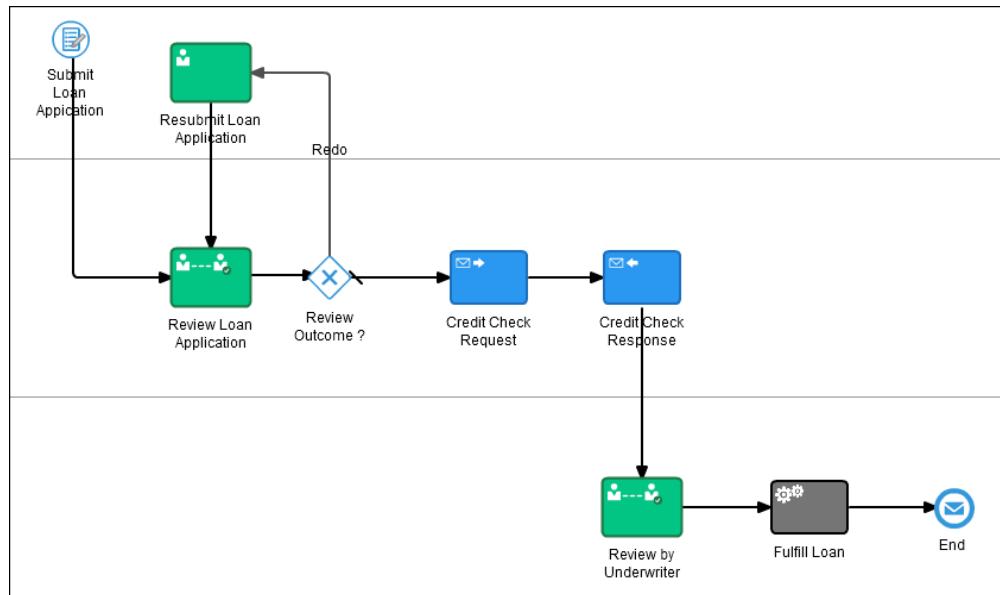
Flow Element Types	Description
Tasks	Represent the work performed by a business process.
Events	Define something that happens during a business process.
Gateways	Determine the flow of your business process.
Sequence Flows	Connect flow elements.

Data Objects

While flow elements 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 business process. A process instance uses these variables to store specific information. At runtime, the business process instance generates and stores specific values for these variables.

Example of a Loan Application Process

Let's look at a sample loan application process. It provides a real-world example of many Process features.

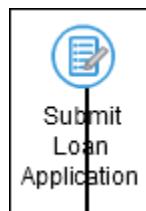


This sample process can be broken down into the following high-level tasks:

- Submit Loan Application
- Determine Application Review
- Check Credit Rating
- Approvals Outcome

Submit Loan Application

The initial form start event is used to trigger a process instance when a user submits a form.



The Submit Loan Application portion performs the following tasks:

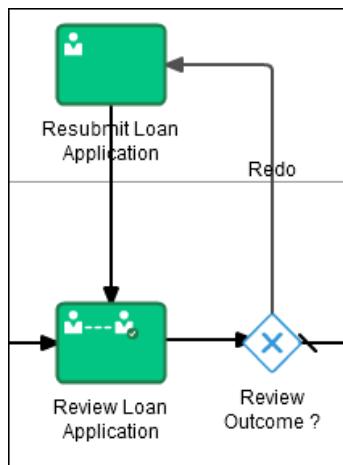
- Defines the start point form start event.
- Initiates the process instance when a user submits a form.

The form is specified in the implementation for the form start event.

Determine Application Review

The next set of flow elements determine if the loan application is complete. After the user enters information, the process flow passes the outgoing sequence flow to the approval human task. If the loan application is approved, then the flow passes to the

check credit rating service element. If the loan application is rejected, then the flow passes to the re-submit human task for more information from the submitter.



Determining the application review:

- 1. Determine approval flow (approve task).**

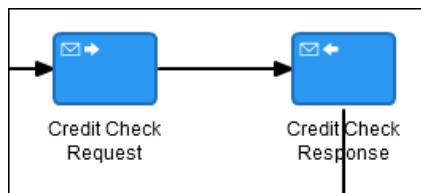
This stage begins with an Approve task, which lets you display a form that the user must review and then perform a certain action. The user might approve or reject the application.

- 2. Check approval flow (exclusive gateway).**

- If Reject, resubmit loan application (submit task).
- If Approve, proceed directly to the check credit rating stage.

Check Credit Rating

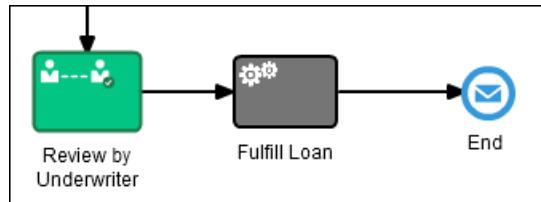
The send task is paired with the receive task to invoke a business process and receive a response in return.



In this example, a message is sent to a business process outside of the current process. After this message is sent, the task is complete and the running of the business process continues. In contrast to the send task, the receive task waits for a message from the business process, which is outside the current business process. After this message is received, the task is complete and the running of the business process continues to the approvals outcome stage.

Approvals Outcome

The approvals outcome stage represents the final stage of the loan application business process. It begins with a review by the underwriter. If the loan is approved, then the process proceeds to the final process flow, which proceeds to the end event.

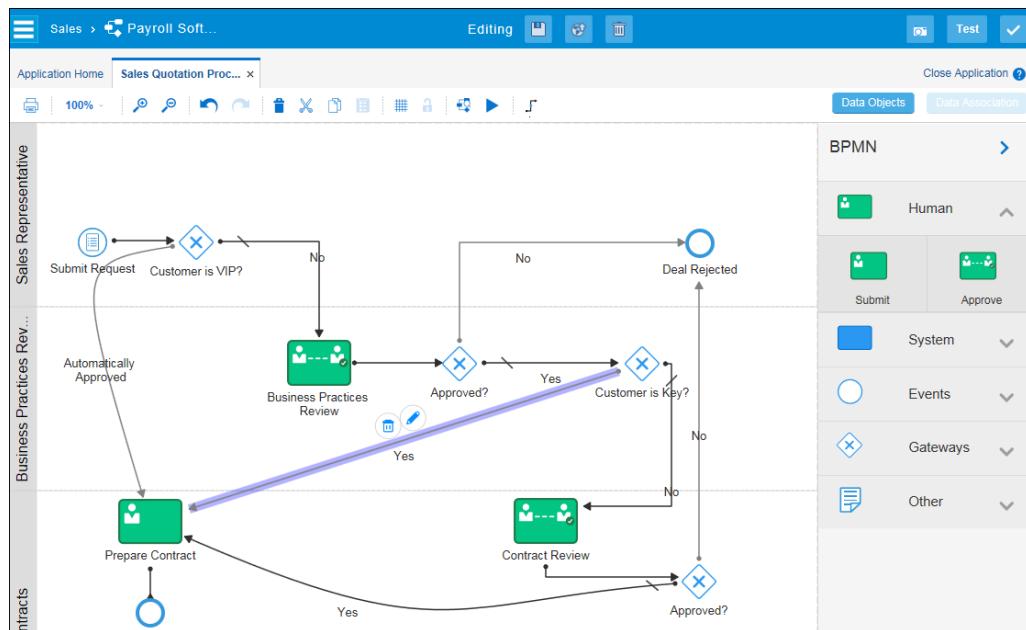


What You Can Do Using the Process Editor

The process editor provides tools for you to easily create and edit business processes.

The process editor opens as a tab and therefore, the toolbar is visible. The options on the toolbar apply to the application as a whole and are available when you're editing a process.

The process editor is divided into three areas: *Toolbar*, *Canvas*, and *Elements Palette*.



Process Editor Toolbar

The process editor toolbar provides quick access to controls related to process design.



Toolbar Icon	Name	Keyboard Shortcut	Description
	Print	Not available	Print your process using the printer setup of your browser.
	Set View	Not available	Set the view size.
	Zoom In	Shift + +	Change to a more close-up view of the process flow.
	Zoom Out	Shift + -	Change to a more distant view of the process flow.

Toolbar Icon	Name	Keyboard Shortcut	Description
	Undo	Ctrl + Z	Revert to the last change made to your process.
	Redo	Ctrl + Y	Reverse the last undo action you performed.
	Delete	Delete key	<p>Delete the selected items from your process.</p> <p>When you delete a flow element that contains an incoming and outgoing sequence flow, the incoming sequence flow is automatically connected to the outgoing sequence flow. However, you may need to manually re-configure the surrounding flow elements.</p> <p>When you delete a sequence flow from a business process, any implementation details you may have configured for that sequence flow are lost.</p>
	Cut	Ctrl + X	<p>Cut the selected items and copy them to the clipboard.</p> <p>When you cut a flow element that contains an incoming and outgoing sequence flow, the incoming sequence flow is automatically connected to the outgoing sequence flow. However, you may need to manually re-configure the surrounding flow elements.</p>
	Copy	Ctrl + C	<p>Copy the selected items to the clipboard.</p> <p>Note: When you copy a human task, the associated form is not copied.</p>
	Paste	Ctrl + V	Paste the items currently in the clipboard.
	Show/hide Grid	G	Toggle the display of a grid in the process editor window.
	Snap to Grid	S	<p>Center the flow elements in your process on the closest grid axis.</p> <p>Existing flow elements are automatically centered and new flow elements are automatically centered when they're added.</p> <p>Snap to Grid is active only when the grid is shown.</p>
	Find Usages	Not available	Display which other processes within the current application calls your process.
	Play Process	Not available	Test the process using the application player.
	Correlations	Not available	Create correlation keys and properties, and then associate one or more properties with each key. Use the correlations to enable business processes to communicate with each other.

Toolbar Icon	Name	Keyboard Shortcut	Description
	Change Icon	Not available	Change the icon of the selected item. You can also click it to return to the default. Change Icon is available only when a valid flow element is selected.
	Change Type	Not available	Change the type of the selected item. Change Type is available only when a valid flow element is selected.
	Change Sequence Flow Type	Not available	Change the type of the selected sequence flow. For example, you can change the sequence flow from a straight line to a curved line. Change Sequence Flow Type is available only when a sequence flow element is selected.
	Open Data Objects	D	Define your data. You can create new data objects based on the business types that are defined for your application.
	Open Data Associations	A	Link data to the activity inputs and outputs.

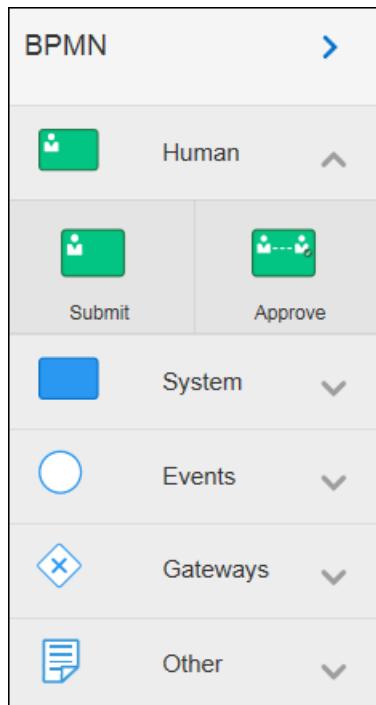
Process Editor Canvas

The process editor canvas is the central area of the process editor window. Use the canvas to draw a diagram that represents your business process using the elements available in the Elements Palette.

The horizontal lines that run across the canvas are swimlanes. Click **New Swimlane**  to add a new swimlane. See [Working with Process Roles and Swimlanes](#).

Elements Palette

Use the Elements Palette to add flow elements and sequence flows to your business process by dragging and dropping elements to the process editor canvas.



Click **Expand** to expand a specific type of palette item and see all the elements available for that type. Click the icon again to collapse. Click **Show/Hide Palette** to close and open the palette. See [Working with Elements](#).

Typical Workflow for Creating a Business Process

Use this typical workflow to create a business process.

Task	Description
Create a process	Create new business processes from the Application Home tab.
Assign roles to the process	Assign roles to swimlanes and determine which members of your business organization are responsible for completing the tasks and activities within the business process.
Design the flow	Drag and drop elements , such as human tasks, system activities, and other events, onto the canvas.
Configure flow element properties	Select a flow element in your process diagram, click Menu , and select Open Properties to open the Implementation pane. Configure properties specific to the flow element. For example, configure a form, service call, or send task and receive task.
Define the data	Define the data the activity uses.
Associate the data	Link data to the activity inputs and outputs.

Creating a Business Process

A business process contains a start event, an end event, and possibly other flow elements. You can start your process from scratch (that is, with an empty start event),

with a form, or when a message is received. Alternatively, you can select a pre-defined process pattern to begin creating a new process.

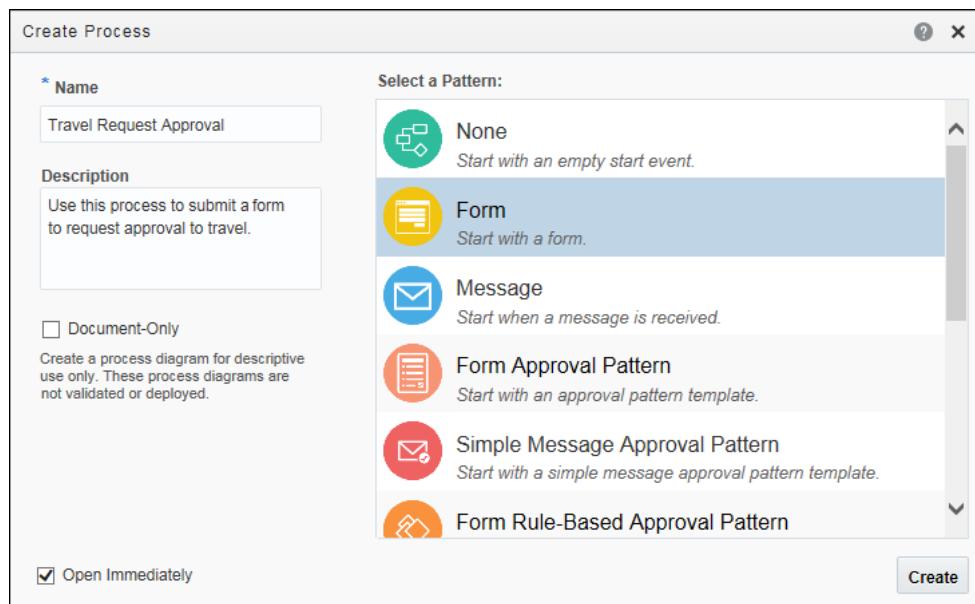
Pre-defined process patterns provide a simplified version of the most commonly-created processes. Because it's easier to edit rather than to create, a pre-defined process supplies you with a shortcut to creating processes, and a way to test other features in Process that require a complete process.

 **Note:**

If you create an application based on a QuickStart App, then a submit and approve business process is automatically created. You can use this process to either play and activate to a test environment to learn more about the functionality of the design-time environment, or you can use it as a starting point to build your own processes.

To create a business process:

1. On the Application Home tab, click **Processes** in the Components pane.
2. Click **New process**  to open the Create Process dialog box.



3. Enter a name for the process.

The field is populated with a default name, such as *Process*, *Process1*, and so on. You can use this name or change it later.

 **Note:**

To rename an existing process, open the process, click **Restore Pane** , click **Business Properties**, select the **General** tab, and then update the process name.

4. Select a pattern from the list. You can start with **None** (empty start event), **Form**, **Message**, or a pre-defined process pattern.
 - If you start your process with a Form, the form isn't created at this point. Instead, you're only setting up the process to expect a form that must be built and associated with the form start event.
 - If you start your process using a pre-defined process pattern, you must still add the implementation details for every flow element that has been automatically added.
 - If you want to start your process with a document or folder, select **None**. You then need to manually adjust the process by replacing the none start event with the **Document Start** or **Folder Start** event. See [Creating a Document- or Folder-Initiated Process](#).
5. Review the optional settings and make your changes, if any.
 - In the **Description** field, enter helpful information about this process, what it's all about, and why you would use it.
 - If this process is for descriptive use only, select the **Document-Only** check box.

Business processes often include details that are necessary for the process to run, which can complicate the picture for someone who only wants to understand what the process does at a high-level. A document-only process provides a simplified process diagram for descriptive use. Document-only processes aren't validated and can coexist with deployable processes in applications.
 - If you want to create the process but not edit it right now, deselect the **Open Immediately** check box to return to the Application Home tab. For example, you may want to create several processes at once before you edit any of them. You can select and edit your process at any time.
6. Click **Create**.

Deleting a Business Process

You can delete one or more business processes from your application at any time.

Before you delete a business process, make sure the process isn't referenced elsewhere in your application. For example, if the process you want to delete is invoked from another business process through a message throw event, then you'll need to re-configure the invoking process to refer to a different process.

An error is displayed during validation if any remaining references to the deleted business process exist.

To delete a process:

1. Go to the Composer page and open an application.
2. Click **Processes**.
3. Find the process you want to delete and click **Delete** .
4. Click **OK** to confirm you want to delete the process from the application.

Importing a Process Created Externally

In Process, you can import external process models created in other programs.

Process supports importing and converting process models in the following formats:

- Visio (vdx)
- XPDL 2.x (xpdl)

It may be necessary for you to modify Visio and XPDL processes before conversion to make sure they're converted accurately.

Note:

If the original file contains properties and artifacts that aren't supported, the unsupported elements aren't converted and are omitted from the final process.

To import a process model:

1. Open your process application.
2. Click **Menu**  in the toolbar and select **Import Model** to open the Process Model Converter wizard.
3. Click **Next**.
4. Highlight the format type of the file you want to import and then click **Browse** to find the file.
5. Select one of the following choices if you're importing a Visio or XPDL file:
 - Create a separate model from each pool
 - Merge pools into a single modelThis dialog box opens even if the original file doesn't contain multiple pools.
6. Click **Finish**.

Working with Process Roles and Swimlanes

Roles are assigned to swimlanes and determine who or what in your business organization is responsible for performing the work of your process-based application.

The key to designing a business process is to determine the people and roles required to complete each task that requires user interaction. Some tasks may be performed by any user or handled automatically by the system. You use swimlanes to group flow elements based on the roles defined within your business process.

Using Roles to Define Responsibilities

Roles are used to decide who or what is responsible for performing the work that is performed within your business processes. Roles allow you to define functional categories that represent job functions or responsibilities within your organization.

The roles defined in your business process are also referred to as logical roles. When your application is deployed to the runtime environment, these roles are mapped to LDAP roles that correspond to the users in your real-world organization.

You can create and edit the required roles responsible for completing activities and tasks within your business process and assign them to the horizontal swimlanes.

As a process analyst, you're responsible for determining what roles are required when designing a business process. For example, for a loan approval business process, you may define the following roles:

- **Loan Officer:** Loan officers are responsible for creating the loan request and forwarding it to the loan processor.
- **Loan Processor:** Loan processors are responsible for reviewing the loan application and checking the credit history of the loan applicant before forwarding the application to the underwriter for approval.
- **Underwriter:** Underwriters are responsible for reviewing and approving the loan application. Additionally, they disburse the loan if it's approved.

Swimlanes and Flow Elements

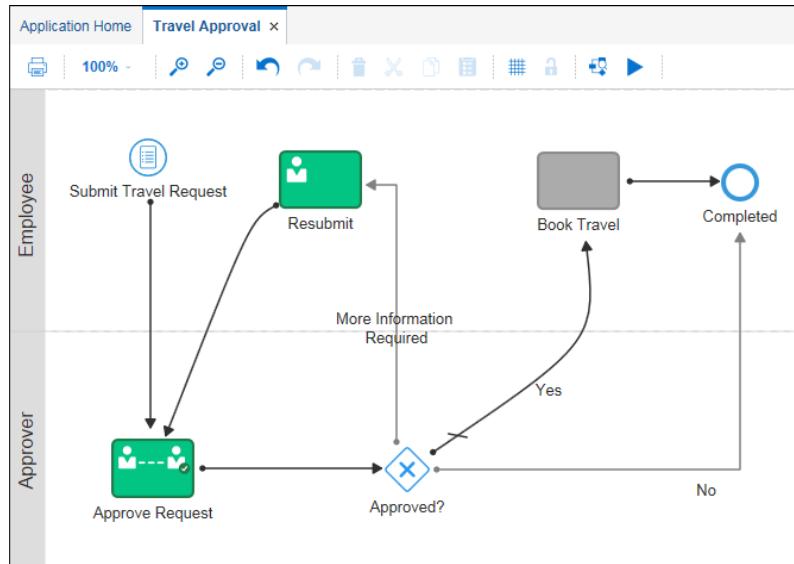
Swimlanes are the horizontal lines that run across the process editor canvas. All flow elements must be placed within a swimlane.

Swimlanes that contain user tasks must have roles assigned to them. Swimlanes visually display the role responsible for performing each flow element within your business process. Additionally, you can have multiple swimlanes that are assigned to the same role. You can also explicitly define a different role for a specific flow element if it needs to be different from the roles assigned to the swimlanes.

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

When you create a new business process, a default swimlane is created. You can add additional swimlanes to your business process as necessary. When adding interactive and manual activities to a business process, you must assign a role to the swimlane.

As shown in the following illustration, the process editor canvas is divided into swimlanes. Each swimlane displays the role of the person responsible for performing the tasks displayed within each swimlane. For example, for a travel request approval business flow, you may have two swimlanes. The *Employee* role is assigned to the first swimlane and the *Approver* role is assigned to the second swimlane. The Approve user tasks appears in the second swimlane so process participants assigned to the *Approver* role are responsible for performing these tasks.



Adding and Editing Swimlanes

To add a new swimlane and edit swimlane properties:

1. Open your application and then open the process.
2. Click **Create New Lane** at the bottom of the page.
3. Click the role name for the swimlane and then click **Edit**.
4. Assign a role to the swimlane.
 - To assign an existing role to the swimlane, select the role from the drop-down list.
 - To assign a new role to the swimlane, click **Add Role** and enter the name of the new role.
5. Change the background color of the swimlane if you want.

Working with Elements

Use the Elements Palette to add elements to a process by dragging and dropping them onto the process editor canvas. After you add an element, you define its properties, data association, forms, and sequence flows where they apply.

Elements are divided into the following types:

- Human
- System
- Events
- Gateways
- Other

Human

Human flow elements represent tasks where a process participant is required to perform the work. The task can be a simple interaction, such as filling out a form, or part of a more complicated workflow that requires input from multiple process participants.

- A **submit task** provides a form or submit action that the user acts on to create a request or provide information about a certain subject.
- An **approve task** provides a form for review or an approve/reject action that the user acts on to approve or reject the request.

See [Adding Human Interaction](#).

System

System flow elements allow you to define interactions across business processes. For example, you can use a Service flow element to invoke an external service or process, a Call flow element to call a reusable process from within the current process, or a Send flow element to send a message to a system or process outside the current process.

System Task Types	Description
Abstract	Use to designate a placeholder for another activity or task.
Services	Use to communicate with other processes and services.
Notify	Use to send an email notification to a user.
Call	Use to call a reusable process from within the current process.
Send and Receive	Use to send and receive messages to and from a system or business process outside of the current process.
Decisions	Use to incorporate Decision Modeling Notation (DMN) decision model snapshots within your business process.
Rules	Use to control process flow or perform calculations using business rules.
Subprocess	Use to group, embed, segment, or reuse processes.

Events

Event flow elements can be divided into two types:

- Start and End elements that define the starting and ending points of a process
- Intermediate events that can either occur within the typical flow of your process or trigger an interruption with your process

Event Types	Description
Document Start	Use to trigger a process instance when a document is received.
Folder Start	Use to trigger a process instance when a folder is received.
Start (none)	Use when no instance trigger is defined, such as when a process instance is created by another flow element, or as a placeholder when the start event isn't known.
Form Start	Use to trigger a process instance when a user submits a form.
Message Start	Use to trigger a process instance when an email message is received.

Event Types	Description
Message Throw and Catch	Use to send and receive messages to and from another business process or service.
Timer Catch	Use to control the flow of your business process using a time condition.
Error Boundary	Use to handle an error that occurs within a process flow.
End Event (none)	Use to mark the end of a process path, or as a placeholder.
Message End	Use to send a message to another process or service when the process is completed.
Error End	Use when the end of a process is the result of an error condition.
Terminate End	Use to immediately stop a process.

Gateways

Gateway flow elements determine the path a token takes through a process. They define control points within your process by splitting and merging paths.

Gateway Types	Description
Exclusive	Splits a process into multiple paths, where process flow continues down only one of the paths. The decision about which path the process should proceed along is based on data-specific conditions.
Inclusive	Splits a process into multiple paths, where process flow can continue down multiple paths depending on conditional sequence flow.
Parallel	Splits a process into multiple paths, where process flow continues down all paths simultaneously.
Event-Based	Splits a process into multiple paths, where process flow continues down only one of the paths. An event-based gateway is similar to an exclusive gateway because both involve one path in the flow. However, for an event-based gateway, decisions about process flow are based on an event taking place rather than a condition being met.

Other

Notes are equivalent to sticky notes. They're temporary and you should use them more as a reminder and delete them as soon as the information is used.

See [Creating Process-Level Documentation](#).

Adding Elements

You can add a flow element to your business process by dragging it from the Elements palette onto the process editor canvas.

To add an element to a process:

1. In the Elements palette, click **Expand**  next to an element type.
2. Click and drag an element onto the process editor canvas where you want to add it.

The cursor displays the icon associated with the element type.

3. Position the cursor at the point in your process where you want to add the flow element.

 **Note:**

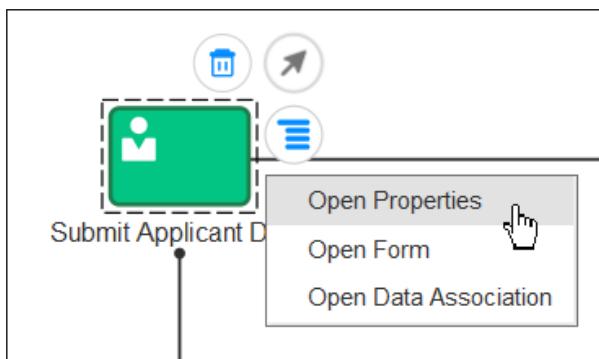
If you position the cursor over a sequence flow, incoming and outgoing sequence flows for the new element are automatically created.

Adding a gateway element, such as an exclusive gateway, requires more steps. See [Creating a Gateway](#).

Editing an Element's Properties

You can edit the basic properties of a flow element using the process editor.

In the process editor, select a flow element, click **Menu** , and select **Open Properties**.



Looking for a keyboard shortcut? After you select a flow element, press **P** to open the element's properties.

Defining Process Start and End

Start events are flow elements that define the starting point of a process. Different types of start events determine how process instances are created. End events, in contrast, define the end point of a process. Different types of end events determine what happens when the process instance is completed.

 **Note:**

In Process, all business processes must have at least one start and one end event.

Because start events define the beginning of a process, they don't have incoming sequence flows. Likewise, end events don't have outgoing sequence flows. However, except for none start and end events, start and end events can have input to and output from other business processes and services.

Specifying the Start and End Events for Different Types of Processes

When you create a new business process, a start event and an end event is created by default. The following table shows the default start and end events depending on the type of process you want to create.

Process Type	Default Start and End Events
None	None start and end event
Form	Form start and None end event
Document	Document start and None end event
Folder	Folder start and None end event
Message	Message start and end event

To learn more, read about the different types of [business processes](#) supported by Process.

Subprocesses contain none start and end events by default. These are the required start and end events and can't be changed. **Event subprocesses** contain a message start event and a none end event by default. However, you can change the start event to reflect the type of event you're handling. See [Working with Subprocesses](#).

Using Multiple Start and End Events in a Process

You can define multiple start points in a business process. Multiple start points allow you to specify multiple ways of creating a process instance, depending on which start event is used. Using multiple start events lets you have multiple ways of starting a process without having to create two separate processes.

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.

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 using multiple end events, it's 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.
- **Terminate end event:** The terminate end event causes all work on a process to stop immediately. No error handling or other cleanup of the running process is performed.

About the None Start Event

Use the none start event when no instance trigger is specifically defined. The none start event can be used as a placeholder when the required start event of a process is unknown, not yet defined, or implemented later by process developers.



None start events also specify the beginning of a process where the process instance is created by another flow element. In general, the none start event doesn't trigger a new process instance.

However, when used with a receive task, the none start event can trigger a new process instance. The receive task must have the `Create Instance` property set to `True`.

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

Note:

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

The none start event doesn't accept process input arguments.

About the Form Start Event

The form start event triggers a process instance when a user submits a form. The form is specified in the implementation for the form start event.



The form is designed to get input from the user and present information relevant to the workflow. As you build the form, a business object is created to store the form data.

About the Document Start Event

The document start event triggers a process instance when document details are received.



The document start event has a defined list of parameters, such as the document ID, type (where "d" stands for document), and document name, that you save in data objects to read the document metadata.

```
"id": "document-id",
"type": "d",
"docName": "document-name",
"startedBy": "user Id",
"role": "contributor"
```

See [Creating a Document- or Folder-Initiated Process](#). You can also use REST APIs to [instantiate a process instance](#) and provide all input values.

Similar to other start events, the document start event can't have incoming sequence flows. Document Start events require a default outgoing sequence flow.

About the Folder Start Event

The folder start event triggers a process instance when folder details are received.



The folder start event has a defined list of parameters, such as the folder ID, type (where "f" stands for folder), and folder name, that you save in data objects to read the folder metadata.

```
"id": "folder-id",
"type": "f",
"docName": "folder-name",
"startedBy": "user Id",
"role": "contributor"
```

See [Creating a Document- or Folder-Initiated Process](#). You then use REST APIs to [instantiate a process instance](#) and provide all input values.

Similar to other start events, the folder start event can't have incoming sequence flows. Folder start events require a default outgoing sequence flow.

About the Message Start Event

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



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

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

You can expose a business process as a service that allows 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.

The message start event allows 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 [Defining Input or Output Arguments](#).

For a business use case, you can invoke a process from any WebService client, such as a SOAP user interface.

To invoke a process using a WSDL:

1. Go to the Composer page and create an application using the **Start Message Event**.
2. Deploy the applications and go to the **Management** page.

3. From the list of deployed application, select the application you just deployed.
4. Click **Actions** and select **WebService**.
5. Copy the **url** displayed in the **Exposed Webservice** tab.
6. Open a third party WebService client, for example, the SOAP UI,
7. Create a new project, enter a **Project Name** and enter the copied **url** in the **Initial WSDL** field, and then click **OK**.
8. Click **Start Operations** and create a basic authorization.
9. Enter the **Username** and **Password** for authorization.
10. Click the **Play** button on the top.

The process is invoked.

About the None End Event

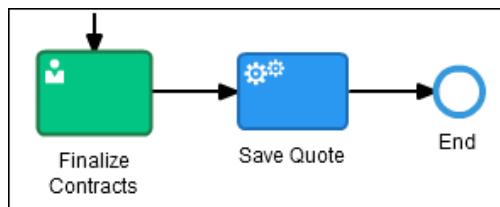
The none end event marks the end of a process path. When a token reaches a none end event, it's consumed. If there are no other tokens within the process instance, then the instance is complete.



Use the none end event:

- When your process isn't required to perform any action after it completes
- To always mark the end of a subprocess and an event subprocess
- As a placeholder to be changed later during implementation by a process developer

For example, in a Sales Quote business process, the Sales Quote service task saves information about the sales quote to a database. 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.



About the Message End Event

Use the message end event to send a message to another process or service when the process is completed.



You always use the message end event with either a message start event or message catch event.

When creating a process that has multiple end events, make sure that any tokens that reach a message end event were created by a message start event. For example, you can't use a message end event to end a process instance initiated by a form start event.

Want to learn more about how to configure output arguments using message end events? See [Defining Input or Output Arguments](#).

About the Error End Event

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



You typically use the error end event 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 with an error end event. See [About the Error Boundary Event](#).

About the Terminate End Event

Use the terminate end event to immediately stop a process.



The terminate end event causes all work on a process to stop immediately. No error handling or other cleanup of the running process is performed.

Adding Human Interaction

Use human tasks to model the user interaction with the application. In Process, process participants interact with your business application during runtime.

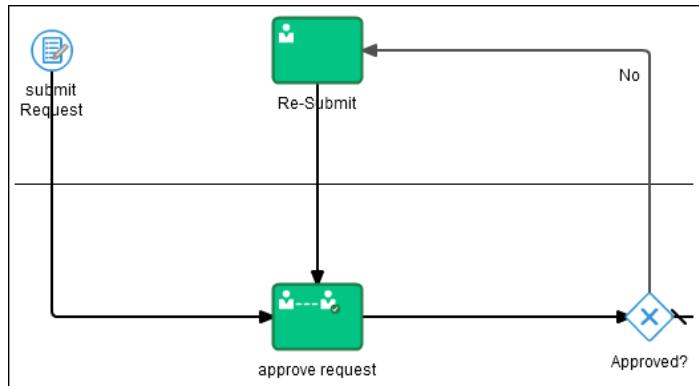
The human task represents a part of your process where a process participant is required to perform an action. The task can be a simple interaction, such as entering a form, or part of a more complicated workflow that requires input from multiple process participants.

After you create the human task, you can configure implementation details such as its assignment, priority, due date, reminders, and deadlines. Human tasks may also contain incoming and default outgoing sequence flows.

Different types of human tasks let you model different types of interactions.

Human Task	Description
A green rounded rectangle containing a white icon of a person with a checkmark.	Lets you display a form that the user must submit to create a request or to provide information about a certain subject.
A green rounded rectangle containing a white icon of two people connected by dashed lines.	Lets you display a form that the user must review or complete, and then perform a certain action. The user might approve or reject the request. You can also define custom actions for the user to perform. Approval tasks lets you define an approval pattern. Generally, you use the outcome of the approval task to drive the rest of the process flow.

When a token reaches a human task, the corresponding task is performed. The token waits until the human task is completed before continuing to the next flow element. For example, look at the following process where the loan request is submitted using the form start event:

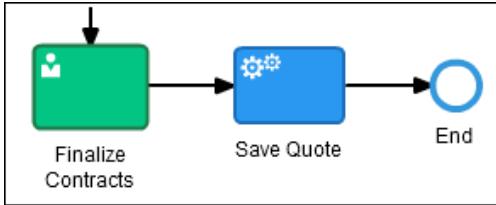


After the user enters information into the form, the process flow passes the outgoing sequence flow to the loan approval human task. If the loan is approved, then the flow passes to the fulfill human task. If the loan application is rejected, then the flow passes to the re-submit human task for more information from the applicant.

See [Working with Human Tasks](#).

Connecting to Other Processes and Services

More likely than not, you'll need to define interactions across business processes within a process-oriented application. For example, your business process may need to invoke an external service, call another process from within the current process, or pass a message to a process outside of the current process. Use these system and event flow elements to model communication between processes and services.

Flow Element	Description
 Service Task	<p>Use the service task to communicate with other processes and services. Add the service task when you know that your business process must invoke an external service or process.</p> <p>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.</p> <p>For example, the following process uses the service task to save the finalized sales quote to a database.</p>  <pre> graph LR A[Finalize Contracts] --> B[Save Quote] B --> C([End]) </pre> <p>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 invokes processes and services synchronously.</p>
 Notify Task	<p>Use the notify task to generate and send notifications. The notify task, which is similar to the service task, uses a predefined service to perform notifications. You use expressions to determine which users receive the notifications generated by the notify task.</p> <p>Currently, email is the only type of notification supported in Process. This type of notification sends an email to the users you specify.</p>

Flow Element	Description
 Call Activity	<p>Use the call activity to call a reusable process from within the current process. The process being called becomes a child process of the calling process.</p> <p>When calling a reusable process, the call activity of the parent process waits until the child process completes before continuing.</p> <p>Data objects of the parent process aren't automatically available to the reusable process. Data objects must be passed to and from the child process using argument mapping of the call activity. See About Reusable Processes (Reusable Subprocesses).</p>
 and  Send and Receive Tasks	<p>Use the send task to pass a message to a system or business process outside of the current process. After this message is sent, the task is complete and the running of the business process continues to the next task in the process flow.</p> <p>You frequently pair the send task with the receive task to invoke a business process or service, and then receive a response in return. The receive task waits for a message from a system or business process outside the current business process. After this message is received, the task is complete and the running of the business process continues to the next task in the process flow.</p>
 and  Message Throw and Catch Events	<p>The send and receive tasks invoke business processes and services asynchronously. If you need to invoke a business process or service synchronously, then use the service task instead.</p> <p>The send and receive tasks perform functions similar to the message throw and catch events. However, you can't use the send task to invoke a business process that is initiated with a message start event.</p> <p>See Using Send and Receive to Communicate Between Processes.</p>
 and  Message Throw and Catch Events	<p>Use the message throw event to send a message to another business process or service.</p> <p>First, you add a message throw event to a business process to define where that business process must invoke another business process or service. You must also implement the connectivity with other business processes, and create and implement the services invoked by the message throw event. See Communicating Between Processes.</p> <p>Message throw events are frequently used with message catch events to receive a response from the business process or service invoked. After the message throw event sends a message to another business process or service, the token immediately moves to the next flow element of the business process. See Using Message Throw and Catch to Communicate Between Processes.</p> <p>The message throw and catch events invoke business processes and services asynchronously. If your process needs to receive a response synchronously, then use the service task instead.</p> <p>You can use a message throw event to invoke other business processes by calling the message start event of another business process. See About the Message Start Event.</p>

Controlling Process Flow Using Gateways

Gateways are flow elements that define the flow of your business process. Gateways determine the path a token takes through a business process. They define control points within your business 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 business process.

Topics

- [Exclusive Gateway: Take Only One Path \(Data Condition\)](#)
- [Inclusive Gateway: Take One or More Paths](#)
- [Parallel Gateway: Take All Paths Simultaneously](#)
- [Event-Based Gateway: Take Only One Path \(Event Occurrence\)](#)
- [Creating a Gateway](#)

Exclusive Gateway: Take Only One Path (Data Condition)

The exclusive gateway lets you split your business process into two or more paths. However, the business process only continues down one of the 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 [Working with Expressions](#).

Evaluating the Flows of an Exclusive Gateway

When a process token reaches an exclusive gateway, the outgoing sequence flows are evaluated until one of them evaluates to *True*.



You can define the order in which the flows are evaluated by configuring the properties for the gateway.

- If only one outgoing sequence flow evaluates to *True*, then the process token continues down that outgoing sequence flow to the next flow element.
- If more than one outgoing sequence flow evaluates to *True*, then the process token continues down the first sequence flow according to the order you defined in the gateway properties.
- If none of the outgoing sequence flows evaluates to *True*, then the process token moves down the default outgoing sequence flow. Therefore, you must define a default outgoing sequence flow for the exclusive gateway.

Unlike other gateways, the exclusive gateway doesn't require a corresponding merge to be explicitly defined in your process after splitting.

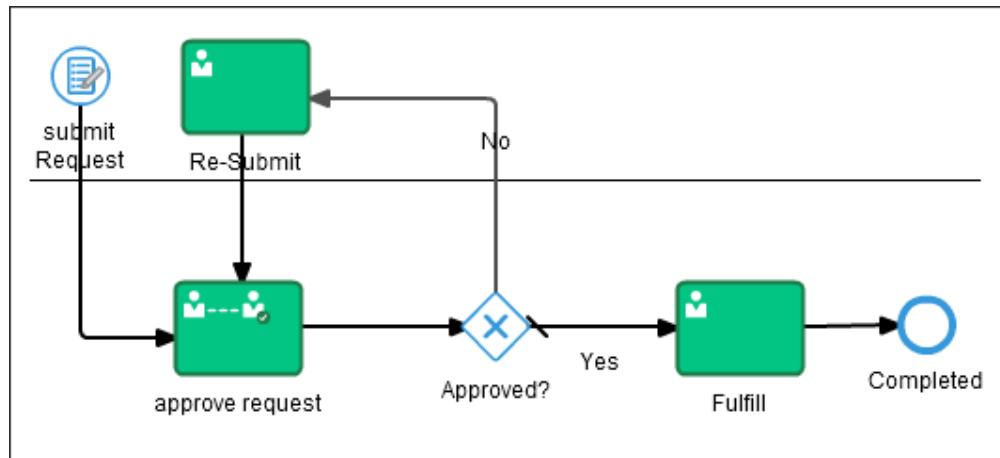
Note:

The exclusive gateway can also merge incoming sequence flows. However, there's no synchronization with other tokens that may be coming from other paths within the process flow. If other tokens arrive at an exclusive gateway merge, then they're passed through as is. If you're synchronizing tokens or performing evaluations on incoming sequence flows, then you should use a different type of gateway.

Example of an Exclusive Gateway

The following diagram shows an example of the exclusive gateway used within a Loan Approval business process. Here, the exclusive gateway is used to evaluate if the loan request is approved or if more information is required.

This evaluation is determined by the expression defined for the outgoing conditional sequence flow. If the expression evaluates to true, then the process flow proceeds down the No path. If it evaluates to false, then the process flow proceeds down the path of the default outgoing sequence flow.

**Inclusive Gateway: Take One or More Paths**

The inclusive gateway lets you split your business 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.

The inclusive gateway requires a split-merge pair. When you add an inclusive gateway to your business process, the split and merge flow objects are automatically created:



The merge portion of the gateway is required. However, you don't have to ensure that all paths out of the split return to the merge.

Although it's possible to have process paths that split at a gateway without merging through the gateway, it's not a good practice and not recommended.

 **Note:**

If you delete the merge gateway from a business process, the corresponding split gateway is also deleted.

Splitting and Merging Inclusive Gateways

The inclusive gateway splits a business process similar to the exclusive gateway, but allows tokens to proceed down one OR more outgoing sequence flows. You can define any number of outgoing conditional sequence flows for an inclusive gateway split. You must define at least one default sequence flow.

When a token arrives at an inclusive gateway, the expressions of its conditional sequence flows are evaluated.

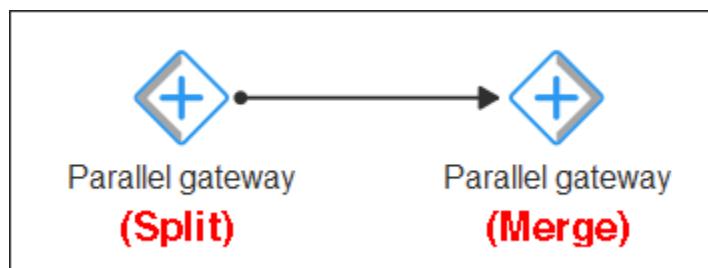
Next, a token is generated for each of the conditional sequence flows that evaluates to *True*. A token is generated for the default sequence flow only if none of the conditional sequence flows evaluate to *True*.

These tokens are joined at the merge of the inclusive gateway. When a token reaches the merge gateway, it waits until all the tokens generated by the split have reached the merge. After all 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.

Parallel Gateway: Take All Paths Simultaneously

The parallel gateway lets you split your business process into two or more paths when you want your process flow to follow all paths simultaneously. The parallel gateway is useful when your business process must perform multiple tasks in parallel.

The parallel gateway requires a split-merge pair. When you add a parallel gateway to your business process, the split and merge flow objects are automatically created:



The merge portion of the gateway is required. However, you don't have to ensure that all paths out of the split return to the merge.

Although it's possible to have process paths that split at a gateway without merging through the gateway, it's not a good practice and not recommended.

 **Note:**

If you delete the merge gateway from a business process, the corresponding split gateway is also deleted.

Splitting and Merging Parallel Gateways

When a token arrives at a parallel gateway, the parallel gateway creates a token for each outgoing sequence flow. The split of the parallel gateway doesn't evaluate any of the outgoing sequence flows.

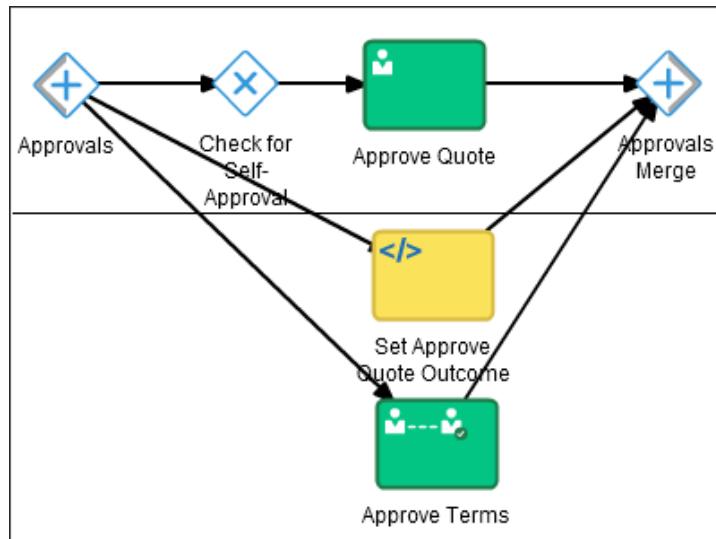
You can also use the parallel gateway to merge process paths split by the parallel gateway. The merge of the parallel gateway waits for a token to arrive from each of the incoming sequence flows. After all tokens arrive, only one token is passed to the outgoing sequence flow.

 **Note:**

You must design your business process so that a token arrives for each incoming sequence flow for the merging parallel gateway. If you don't, your business process can freeze if the merge is expecting tokens that don't arrive.

Example of a Parallel Gateway

In this example, a Sales Quote process uses a parallel gateway at the approval stage. The diagram shows how the parallel gateway is used to execute all process paths at the same time.



Event-Based Gateway: Take Only One Path (Event Occurrence)

The event-based gateway allows you to branch your process flow based on the possibility that an event may occur. Depending on the context, the event may be one of several types.



The event-based gateway allows you to anticipate the possibility that several types of events may occur at a specific point in your process. It's similar to the exclusive gateway, but instead of selecting a path based on data-specific conditions (expressions), the event-based gateway selects 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 no stock is currently available. The process may have to wait until stock is available, but can't 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).

Target Events for Event-Based Gateways

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. For an event-based gateway, you define two or more of the following target events:

- **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 can't 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 can't have sequence flows from other parts of the process. Although the event-based gateway allows you to plan that multiple events may occur in your process, only one event is triggered within the process instance. When the first event in the event-based gateway is triggered, the path that comes after that event is followed.

When you add an event-based gateway to a process, no associated target events are created. When you delete an event-based gateway, any outgoing sequence flows are also deleted. The associated events aren't deleted.

Process Start 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 allow an event-based gateway to create a new process instance:

- You must enable the **Instantiate** property of the event-based gateway.
- You can't have any incoming sequence flows to the event-based gateway.
- Your process must have at least one other start event, for example a message start event, in addition to the event-based gateway.

Although the event-based gateway can be used to create a new process instance, it doesn't accept data input from another process. Any data that must be passed to the process instance must be configured using the target events.

Creating a Gateway

Use gateways to control how the process flows.

To create a gateway:

1. Open your process.
2. In the Elements palette, click **Gateways** to expand the list.

Here are the gateways you can add to a process:

Gateway Icon	Gateway Type	Description
	Exclusive	Only one of the paths out of the gateway is taken. The decision about which path the process should proceed along is based on data-specific conditions. For example, an exclusive gateway can specify different paths for the APPROVE and REJECT outcomes of an Approval human task.
	Inclusive	One or more paths out of the gateway can be taken, and the paths must converge later in the process. Use this type to perform several optional or conditional tasks at the same time.
	Parallel	All paths out of the gateway are taken, and the paths must converge later in the process. Use this type to perform several required tasks at the same time.
	Event-Based	Only one of the paths out of the gateway is taken. An event-based gateway is similar to an exclusive gateway because both involve one path in the flow. However, for an event-based gateway, decisions about process flow are based on an event taking place rather than a condition being met.

3. Drag and drop the gateway element into the process flow.

4. Select the gateway element. Click **Sequence Flow**  and drag the icon to create each path that exits from the gateway.
 - For exclusive or inclusive gateways, make sure the default path connects to the correct flow element. Default sequence flows represent the path your business process takes out of these gateways when none of the data conditions evaluate to true. Default sequence flows are represented by an arrow with a slash mark on one end.
 - For inclusive or parallel gateways, you can drag paths to the side to separate them.
 - For parallel gateways, you must drop at least one activity onto each path before you can create another path.
5. For exclusive or inclusive gateways, create a data object. Data objects store and organize data the process uses.
 - a. Select the gateway and click **Data Objects** in the toolbar. The Data Objects dialog box opens.
 - b. Click **Add**.
 - c. Type a name for this data object. All names must begin with a lowercase letter.
 - d. Select the data type that matches the data flowing into the gateway from the previous task.

In most cases, the type is **boolean** (true or false) or **string** (text). For example, select **string** for the APPROVE and REJECT outcomes of an Approval human task.
 - e. Click **Create**.
 - f. Close the Data Objects dialog box.
6. For exclusive or inclusive gateways, implement each non-default path.

You don't need to implement the default path, but you should name it.

 - a. Select a non-default path, click **Edit**  on the path, and click **Implementation**.
 - b. Type a name for this path.
 - c. Click **Edit**  next to the Condition field.
 - d. Type an expression that tests whether the Gateway data object has one of the choices.

For example, type `approvalOutcome == "APPROVE"`. Note the double equal sign.
 - e. Click **OK**.
 - f. Close the Implementation pane.
7. For exclusive or inclusive gateways, associate the output data with the gateway data object.

A parallel gateway doesn't need data association, but some of the activities on the paths might need it.

- a. Select the task that precedes and sends data to the gateway, and click **Data Association** in the toolbar.

- b. Expand the Process and Data Objects nodes in the right pane.

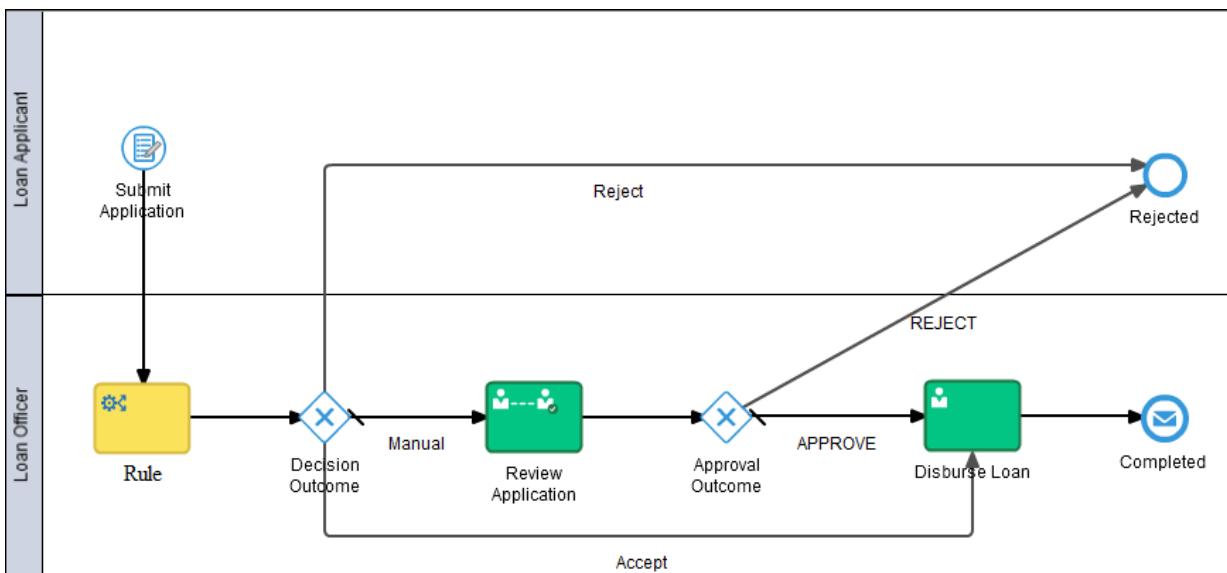
- c. Drag and drop the gateway data object into the Outputs field.

You can associate inputs as well, which often come from a form. If the form is referenced in the start event or another process component, then the form data object appears in the right pane. Drag and drop each form field from the right pane to the corresponding Inputs field.

- d. Click **Apply**.

The Data Association page closes.

8. Click **Save**  in the toolbar to save your changes.



Working with Decisions

Decision models automate operational decisions inherent in your business processes.

Use the **Decision** element to add decision model snapshots to your applications.



You can use different types of decision logic, such as decision tables, simple expressions, and so on, to create an executable model and incorporate it into your business process to facilitate automated decision-making.

Want to know more about working with decision models? See [Creating Decisions](#).

Working with Rules

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

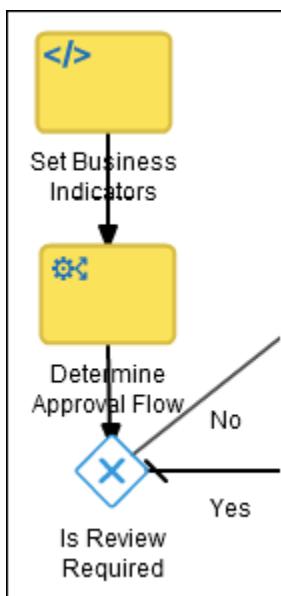
Use the Rule element to incorporate Oracle Business Rules within your business process.



Here are the primary use cases:

- Use **structural rules** to perform calculations used within your business process. For example, you can use a structural business rule to calculate a credit score.
- Use **operative rules** to make changes to the flow of your business process. A typical use of an operative rule is to perform a check of the rule conditions. Then, as part of the output data association, assign a value to a data object using an expression. In this scenario, the decision task is immediately followed by a gateway that is used to branch the process path according to the value of the data object.

The following example shows a rule within a Sales Quote business process. The approval review is determined based on the business rule and data object value.



See [Creating Business Rules](#).

Working with Sequence Flows

Sequence flows define the order or sequence that work is performed within a business process. Sequence flows connect the flow elements within your business process and determine the path a process token follows through your business process.

Incoming sequence flows are the sequence flows that lead into a flow element.

Outgoing sequence flows are the sequence flows that determine the process path out of a flow element.

Most flow elements 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 don't have either incoming or outgoing sequence flows.

To add sequence flows to your business process:

1. In the process editor, click the flow element from where you want to create the outgoing sequence flow.
2. Click **Add Sequence Flow**  and keep the mouse depressed. This option only appears for flow elements that don't already have outgoing sequence flows.
3. Move the cursor to the flow element you want to connect to, and then release the click.

About Unconditional Sequence Flows

Unconditional sequence flows represent the typical path between two flow elements. Default sequence flows are displayed as arrows:



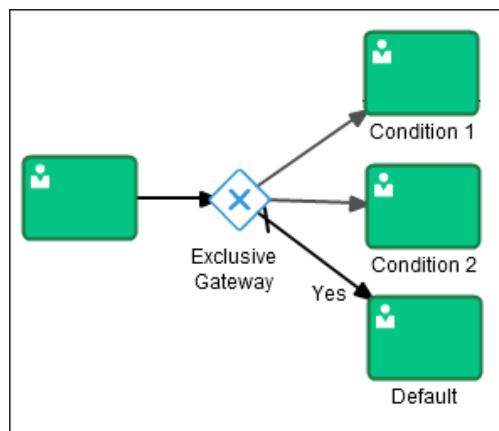
Most flow elements can contain only one default outgoing sequence flow. Only parallel gateways can contain multiple unconditional sequence flows, which represent the parallel paths of your business process.

Exclusive, event-based, inclusive, and conditional gateways can't have unconditional outgoing sequence flows. These gateways use conditional and default sequence flows to determine the flow of your business process.

About Conditional Sequence Flows

Use conditional sequence flows to control the flow of a business process based on certain conditions. Like unconditional sequence flows, conditional sequence flows are represented by an arrow.

Here's an example of two outgoing conditional sequence flows and a default sequence flow (which includes a slash mark at the beginning of its arrow):



Not all flow elements can use outgoing conditional sequence flows. Only the following types of gateways can have outgoing conditional sequence flows:

- Exclusive gateways

- Event-based gateways
- Inclusive gateways (split)

You use [expressions](#) to define the conditions used within a conditional sequence flow.

About Default Sequence Flows

Like conditional sequence flows, default sequence flows are used as outgoing sequence flows to exclusive, event-based, inclusive, and conditional gateways.

Default sequence flows represent the path your business process takes out of these gateways when none of the conditions evaluate to true. Default sequence flows are represented by an arrow with a slash mark on one end.

Working with Intermediate Events

Unlike start and stop events, intermediate events occur during the flow of your business process. You can use them to control the flow and behavior of your business process.

There are two types of intermediate events:

- **Normal flow events:** These events occur within the typical flow of your business process.
- **Boundary events:** These events trigger an interruption with your business process.

Boundary events are associated with flow elements and can be configured to interrupt their usual behavior. They behave similar to sequence flows in that they're used to determine the path a process takes between flow elements.

Boundary events can be divided into two types: interrupting and non-interrupting.

About the Timer Catch Event

Timer catch events lets you control the flow of your business 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

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 process flow the timer event defines. The process flow that an interrupting timer event defines 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 process flow the timer event defines. The process flow that a non-interrupting event defines can't return to the main process flow.

About the Error Boundary Event

Error boundary events are intermediate events used to handle an error that occurs within your process flow.

Error boundary events can be attached to the following flow elements:

- Service tasks
- Call activities
- Human tasks
- Send tasks
- Receive tasks
- Script tasks
- Decision tasks
- Subprocesses

Error boundary events are always interrupting, meaning that they interrupt the usual flow of a business process.

The following image shows the default notation for the error boundary event attached to a service task.



When a service or process fails with an error, the error boundary event is triggered. This causes the process flow to follow the path of the outgoing sequence flow of the error boundary event.

You can use this flow to define how to 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 can't return to the main process 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.

Working with Draft Processes

Process enables you to create and deploy draft processes. However, they must be valid before they can be deployed.

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

When a flow element is marked as draft, you can't configure data associations for it. If you mark a flow element as draft that has previously had data associations configured, these are lost.

You can define the implementation details of a draft flow element. However, it's not required. A draft flow element with no implementation defined doesn't generate errors during validation.

When an application containing a draft flow element is activated, implementation details for those flow elements are ignored. For example, if your business process contains a user task marked as draft, the runtime engine doesn't create instances of the associated human task.

To mark a flow element as draft, select the required flow element, click **Menu** , and select **Open Properties**. Select the **Is Draft** check box.

Marking your process as draft is different than marking it as document-only. Document-only processes are for descriptive use only and aren't used to control the application.

About the Abstract Flow Element

The Abstract flow element isn't an implementable activity. It's just a modeling placeholder for another activity or task.



You can mark the Abstract flow element as a draft element, which means you can deploy your business process and test the parts of your process that have been completed without having to wait until all the flow elements have been completed.

Working with Subprocesses

You can use subprocesses to organize your business processes. Subprocesses lets you group functional areas of your business process together and also make your business processes more readable.

Process supports the following types of subprocesses:

- Reusable processes
- Embedded (also called inline) subprocesses
- Event subprocesses (also called inline handlers)

In addition, you can also configure multi-instance markers in subprocesses.

About Reusable Processes (Reusable Subprocesses)

Process supports reusable processes, sometimes referred to as reusable subprocesses. Reusable processes let you create business processes that can be called from other business processes.

For example, all your business processes may have to charge a credit card, so you can create a charge credit card reusable subprocess.

Reusable processes have the following characteristics:

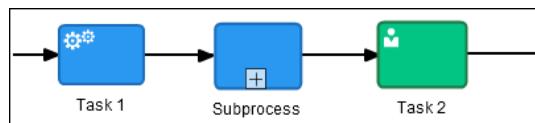
- Must start with one none start event.
- Can contain multiple end events.
- Can only be called by other business processes.

About Embedded Subprocesses (Inline Subprocesses)

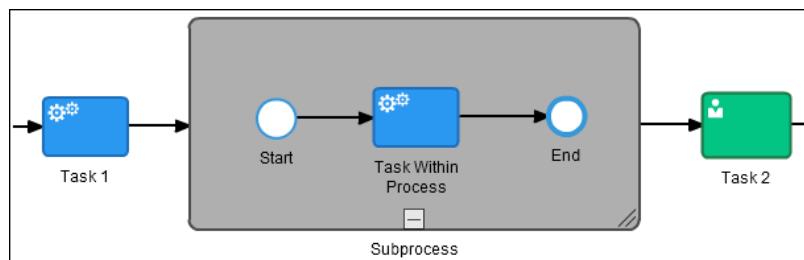
Embedded subprocesses let you group flow elements together to make your business process more readable. In Process, subprocesses are embedded subprocesses.

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. The following is an example of how a collapsed subprocess appears within a process flow.



The following diagram shows how an expanded embedded subprocess appears within a process flow. When an embedded subprocess is expanded, you can edit the flow elements within it. You can also click and drag the edge of the embedded subprocess window to change its size.



Similar to other types of business processes, embedded subprocesses can have 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 can't 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 business process can contain nested embedded subprocesses. However, use nested embedded subprocesses only when necessary to make your business process more readable.

 **Note:**

Creating data objects for an embedded subprocess isn't supported.

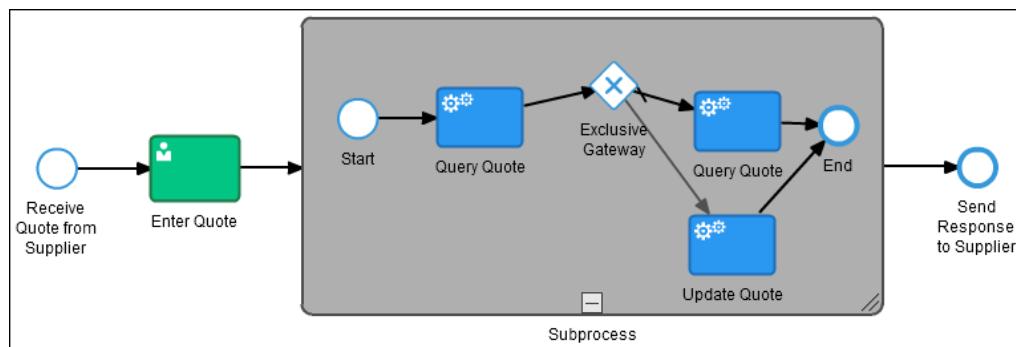
Embedded Subprocesses and Sequence Flows

The flow elements within an embedded subprocess can't have sequence flows that connect to flow elements outside the subprocess.

Similar to other flow elements, embedded subprocesses have incoming and outgoing sequence flows.

Embedded Subprocesses in Context

The following diagram shows an example of an embedded subprocess that groups the service tasks used to process a sales quote.



About Event Subprocesses (Event Handlers)

An event subprocess is a type of subprocess that lets you model possible conditions that happen outside of a normal process flow. Event subprocesses are also called event handlers.

Event subprocesses must begin with a message start event.

Event subprocesses can only contain one start event. However, you can have multiple event handlers within your business process.

Configuring Multi-Instance Markers in Subprocesses

Multi-Instance markers enable you to run a subprocess for each of the elements on a set of data. When a subprocess with a multi-instance loop marker runs, it creates a set of instances, one for each element on the set of data. You can configure the multi-instance marker to process these instances in parallel or in a sequence.

For example, you can use multi-instance markers in repetitive processes, such as creating travel requests for a group of employees, ordering office supplies for a team, and so on. Multiple instances are created, one for each employee, which are then

passed through the process of individual requests being made and forwarded to the approver. These instances can be configured to run either in sequence or in parallel.

If multi-instance markers are used, an indicator for the same appears at the bottom of the subprocess; three horizontal lines indicate sequential execution while two vertical lines indicate parallel execution.



To configure multi-instance markers in a subprocess:

1. Click the subprocess.
2. Select **Properties**.
3. The **Implementation** pane appears.
4. Select **Generate multiple instances** under Repetition Cycle.
5. Select the way in which you want to manage your instances:
 - a. **In Sequence**: Select to specify that each instance must complete the subprocess before the next instance starts to run the subprocess.
 - b. **Parallel**: Select to specify that all instances run in parallel.
6. Specify the type of input you want to supply:
 - a. **Condition**: If your input is not an array, then select **Condition** to specify the number of times to execute the sub-process.
 - i. In the adjacent expression area, enter a number to specify the cardinality. You may also enter a data object or a simple expression to specify the same. To launch the Expression Editor window, click the **Edit Expression** button.
 - ii. In the output expression field, enter the array to store the result of the subprocess.
 - b. **Array**: Select to enter an array of elements as input.
 - i. In the input expression field, enter the array that is to be consumed by the subprocess. You can select a data object or form a simple expression using the Expression Editor window. Generally, the selected data object is a collection of items.
 - ii. In the output expression field, enter the array to store the result of the subprocess.
7. Optionally, you can specify a completion condition to terminate the execution of instances.
 - a. Select the checkbox that reads **Define a condition that terminates execution of subprocess instances**.
 - b. In the resulting expression field, enter a completion condition in terms of a simple expression.

 **Note:**

- At the subprocess level, you can add data objects (of any type) with their scope limited to an individual subprocess execution. Such data objects help you, especially in parallel execution, to keep local copies of data and merge them after all instances are executed.
- In the XPath expression that is used to fetch data objects for each instance, make sure that the index starts from '1' and not from '0'.

 **Important:**

If you have a subprocess that invokes multiple instances of another process, aborting any of these child instances triggered by the parent process will abort all child instances in the same flow.

Predefined Variables

The following predefined variables are available for multi-instance subprocesses. Data communication within a multi-instance subprocess should happen using these variables.

Name	Data Type	Description	Available for MI Array?	Available for MI Condition?
loopCounter	int	The sequence number of the iteration.	Yes	Yes
numberOfInstances	int	The total number of instances.	Yes	Yes
numberOfActiveInstances	int	The total number of active instances.	Yes	Yes
numberOfCompletedInstances	int	The total number of completed instances.	Yes	Yes
numberOfTerminatedInstances	int	The total number of terminated instances.	Yes	Yes
loopDataInput	string[]	The reference of the array that is passed as input to the multi-instance subprocess.	Yes	No
inputDataItem	string	The handle to an element of the <code>loopDataInput</code> array passed as input to the multi-instance subprocess.	Yes	No
loopDataOutput	string[]	The reference of the array that is passed as output of the multi-instance subprocess.	Yes	No
outputDataItem	string	The handle to an element of the <code>loopDataOutput</code> array that is passed as output of the multi-instance subprocess	Yes	No

 Note:

Transformations are not supported for predefined variables.

Communicating Between Processes

Processes can interact using message start and message end events, send and receive activities, or message throw and message catch events. Processes can also call other processes or include subprocesses.

Processes interact *synchronously* when one process starts another and waits for a response. Processes interact *asynchronously* when one process starts another and both processes run independently.

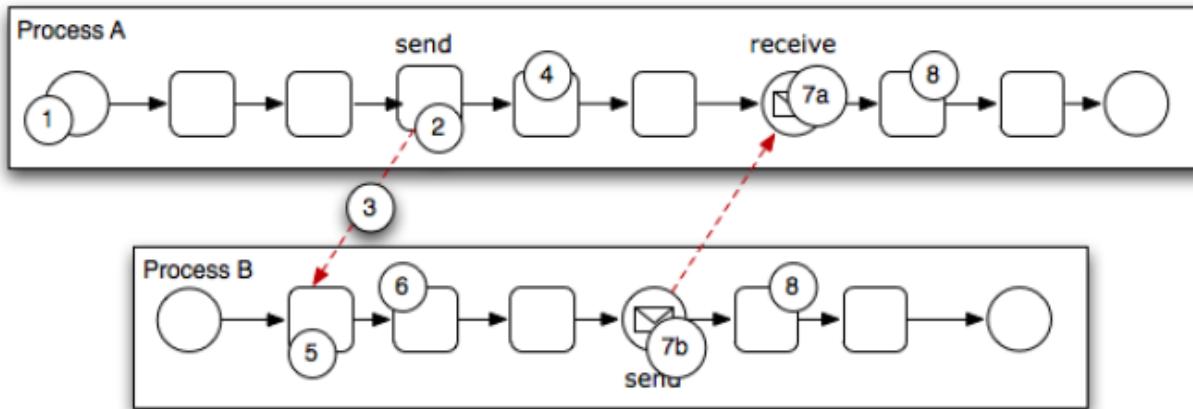
Interaction Type	Description
Message Start and Message End events	<p>If you define input arguments for a process in a message start event, you must supply data for these arguments to start the process.</p> <p>You can asynchronously start a process that begins with a message start event using a message throw event or send activity in another process. Or you can synchronously start such a process using a Service activity.</p> <p>A process that begins with a message start event is exposed as a web service after deployment. Therefore, you can connect with the process as a web service consumer to start it. See Creating a Web Service Connection.</p> <p>If you define output arguments for a process in a message end event, the process supplies data for these arguments when it completes.</p> <p>See Defining Input or Output Arguments.</p>
Send and Receive activities	<p>A send activity sends data to and starts another process. A receive activity receives data from another process when it completes.</p> <p>Send and receive activities can have boundary events, unlike message throw and message catch events, which can't.</p> <p>See Using Send and Receive to Communicate Between Processes.</p> <p>For a send activity in a calling process, see Calling Another Process. For a receive activity or a send activity in the called process, see Defining Input or Output Arguments.</p>
Message Throw and Message Catch events	<p>A message throw event sends data to and starts another process. A message catch event receives data from another process when it completes.</p> <p>See Using Message Throw and Catch to Communicate Between Processes.</p> <p>For a message throw event, see Calling Another Process. For a message catch event, see Defining Input or Output Arguments.</p>
Service activity	<p>A Service activity can call another process or a web service. The send, receive, and call activities and the message throw and message catch events can only call another process asynchronously. The Service activity can call a process asynchronously or synchronously. However, if the called process begins with a message start event, the call must be synchronous.</p> <p>For web service details, see Creating a Web Service Connection.</p> <p>See Calling Another Process.</p>

Interaction Type	Description
Call activity	A Call activity starts another process as a child process. The parent process waits for the child process to complete before continuing. This lets you reuse a process in multiple processes that call it. See Defining Input and Output Arguments for a Child Process .
Subprocess activity	A Subprocess activity contains a subprocess with its own start and end events. A subprocess isn't separate from its parent process. A subprocess activity lets you hide the complex details of a part of a process to make the overall process more readable. Click Expand  to expand the subprocess, then drag and drop activities directly into the subprocess. Click Collapse  to collapse the subprocess.

Using Send and Receive to Communicate Between Processes

Use the send and receive activities to start another business process and receive messages back from it. Processes that begin with a receive activity and contain a send activity are exposed as services that other processes and services within Process can use.

The diagram shows how a send activity starts another process and a receive activity receives a response.



The following steps outline a possible scenario for using send and receive activities to communicate between processes.

1. Process A starts.
 2. The flow of Process A reaches the send activity.
 3. The send activity starts Process B.
 4. The flow of Process A continues to the next flow element.
 5. The receive activity initiates an instance of Process B.
- The send activity implementation specifies which process is started.

The start event in *Process B* must have a Trigger value of None. The receive activity implementation must have **Create Instance** checked.

6. *Process B* runs.
7. Depending on the specific behavior of both processes, the following scenarios may occur:
 - a. If the flow of *Process A* reaches a receive activity paired with a send activity from *Process B*, the flow of *Process A* waits until a response is received.
After the response is received, the flow of *Process A* continues to the next flow element.
 - b. If the flow of *Process B* reaches a send activity paired with a receive activity in *Process A*, *Process B* sends a response to *Process A*.
The flow of *Process B* continues to the next flow element.
8. Both business processes continue running.

You can use subsequent send and receive pairs to define subsequent communication between the two processes.

Using Message Throw and Catch to Communicate Between Processes

Use combinations of message throw and message catch events to start and communicate with other business processes.

When using a message throw event to start another business process, the following conditions must be met:

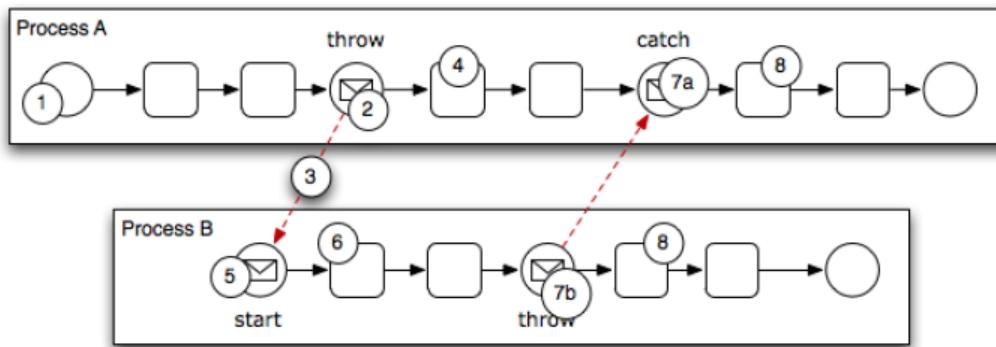
- The business process being started must be an asynchronous process.
Although you can use a message throw event to start a synchronous process, there is no mechanism for catching messages synchronously from the other process. To start a synchronous process, use the Service activity.
- The first time you use a message throw event, it must be paired with the message start event of the process it starts.

This is required to start the process instance. After the instance has been started, you can use subsequent message throw events that are caught by other events in the second process.

Processes that begin with a message start event are exposed as web services. See [Creating a Web Service Connection](#).

Processes started by another process aren't considered child processes. This means that a terminate end event in the calling process doesn't stop a process started with a message throw event.

This diagram shows how a message throw event starts another process and a message catch event receives a response.



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

1. *Process A starts.*
2. *The flow of Process A reaches a message throw event that starts Process B.*
3. *The message throw event sends a message to the message start event of Process B.*
4. *The flow of Process A proceeds to the next flow element.*
5. *The message start event starts an instance of Process B.*
6. *Process B runs.*
7. *Depending on the specific behavior of both processes, the following scenarios may occur:*
 - a. *If the flow of Process A reaches a message catch event paired with a message throw event from Process B, the flow of Process A waits until the message is received.*
After the message is received, the flow of Process A continues to the next flow element.
 - b. *If the flow of Process B reaches a message throw event paired with a message catch event in Process A, Process B throws a message to Process A.*
The flow of Process B continues to the next flow element.
8. *Both processes continue running.*

You can use subsequent message throw and message catch event pairs to define subsequent communication between the two processes.

Defining Input or Output Arguments

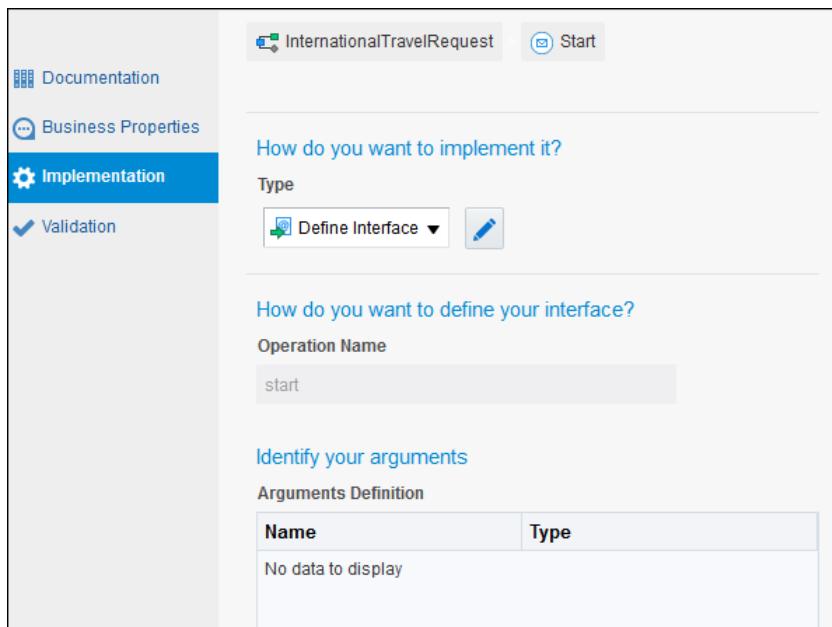
You can define input arguments for message start and message catch events and for receive activities. You can define output arguments for message end events and for send activities in processes called by other processes.

To define input or output arguments for a process interaction event or activity:

1. In the process editor's Elements Palette, expand the system or event elements.
2. Drag and drop the element into the process.

3. Select the element, click **Menu** , and then select **Open Properties**.

The implementation pane appears.



4. For a Receive activity in a process called by another process, check **Create Instance**.
5. Select **Define Interface** from the **Type** drop-down list. Click **Edit**.
6. Click **Add** to add new arguments, then determine the **Name** and **Type** of each argument.
7. For a Send activity in a process called by another process, select **Synchronous** if the process must wait for a reply.
8. Enter an **Operation Name** or accept the default name.
9. Define data associations to pass the input argument data to the next activity in the process. See [Configuring Data Association](#).

When you activate a process that begins with a message start event, the process is exposed as a web service. Information about the web service is displayed when you select **View Deployed Services** on the Manage Deployed Applications page. See [Integrating with Applications and Services](#) and [Administrator Basics](#).

Defining Input and Output Arguments for a Child Process

You can define input and output arguments for a child process called by a call activity in the parent process.

The start and end events in the child process must have Trigger values of None. The child process can't contain elements that only make sense in an independent process, such as message events and send, receive, or service activities.

To define input or output arguments for a child process:

1. Open the child process in the process editor.

2. In the process editor, click the **Restore Pane** icon in the lower-right corner.
The Properties pane opens at the bottom of the window.
3. Click the **Add** icon to add new input or output arguments, then determine the Name and Type of each argument.
4. Open the parent process in the process editor, select the call activity, click **Menu** , and select **Open Properties**.
5. Click **Implementation**.
6. Select the child process from the Process drop-down list and close the Properties pane.
7. Define data associations for the call activity to pass the input data to the child process and receive the output data from it. See [Configuring Data Associations for a Process Flow Element](#).

Calling Another Process

You call another process using a message throw event or a send, service, or call system activity.

To call another process:

1. In the process editor, expand the **System** or **Events** types in the Elements Palette.
2. Drag and drop a system or event element into the process.
3. Select the element, click **Menu** , and select **Open Properties**.
4. Click **Implementation**.
 - For a call activity, select a process to call from the Process drop-down list. Want to learn more about using a Call activity? See [Defining Input and Output Arguments for a Child Process](#).
 - For all other calling components, select **Process Call** from the Type drop-down list. Click the **Browse** icon, select a process, then click **OK**.
5. Select a different Target Node from the drop-down list if necessary.
 - For a message throw event, the message start events in the called process are listed.
 - For a send or service activity, both message start events and receive activities are listed.
6. Define data associations to receive input data from the previous activity in the process. See [Configuring Data Association](#).

Using Correlations to Communicate Between Processes

Are you looking for another way to get your processes to talk to one another? Let's look at how you can develop a process that uses correlations to communicate with other processes and services.

Topics

- [Correlations Lead to Process Communication](#)
- [Components of a Correlation](#)
- [Defining Correlation Keys and Properties](#)
- [Using Your Correlations in a Process](#)

Correlations Lead to Process Communication

Correlations enable business processes to communicate with each other based on the state of an instance. The state of all the process data objects in a process defines the state of the instance.

When you define a correlation for a business process, you can identify an instance in another process based on the instance state and then send a message to that specific instance.

For example, you can use correlations to communicate a sales process with the corresponding shipping and mailing processes. When the customer confirms an order, the shipping process sends a message to the shipping and mailing processes using a correlation that defines that it uses the order ID to locate the instances in both processes.

After you initialize a correlation, you can't change its value because the service engine uses this value to locate the instance.

You can define and initialize multiple correlations for a flow element.

- The flow element that sends the message can use just one correlation or all the correlations defined for that flow element. If the flow element uses all the existing correlations, then all the values it sends together with the message must lead to the identification of the same instance.
- Some flow elements are able to initialize a correlation (that is, Start), some to use it (that is, End), and some to both initialize and use (that is, Receive Task).
- For those activities that manage input and output arguments, such as the service task, you can set the property value based on one or the other.

The scope of the correlation is the instance of the process or subprocess where it's defined.

Be careful not to initialize the same correlation more than once. If you do, you'll get an error at runtime.

Components of a Correlation

The main components of a correlation are definition, keys, properties, and property values.

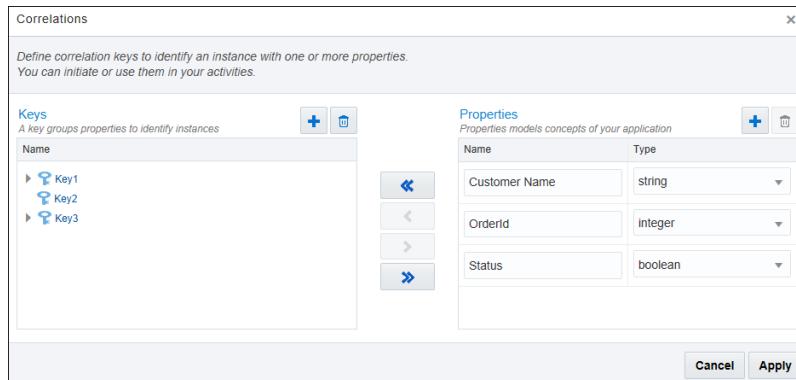
- **Correlation definition** contains the set of correlation keys defined for a flow element.
- **Correlation keys** define a unique name to identify the properties to use in the correlation. When you define a correlation key, you provide a name to identify it. The scope of the correlation key is the process, which means that after you define a correlation key you can use it for the correlation definition of any flow element in that process.
- **Correlation properties** are abstractions for very representative attributes in the process, like the order ID, the customer name, or the social security number. Properties contain a name to identify the attribute and a data type. Properties only support basic data types. The scope of the properties is the application. You can see the property definition in the different processes.
- **Correlation property values** enable you to define how to assign a value to the correlation property using expressions. You can use the arguments and predefined variables of the activity to set the correlation property values.

Defining Correlation Keys and Properties

You define correlation keys at the process level, which means you can reuse correlation keys across your process and flow elements.

To define correlations keys and properties:

1. Go to the Composer page.
2. Open your application.
3. Open your business process.
4. Click **Correlations**  in the toolbar. The Correlations dialog box opens.



5. Create the correlation keys and properties, and then associate one or more properties with each key you create. Use the arrow keys to move information between the Keys column and the Properties column.

Using Your Correlations in a Process

You can define multiple correlations for a single flow element (that is, an activity). The activity that sends the message can use one correlation, several correlations, or all of

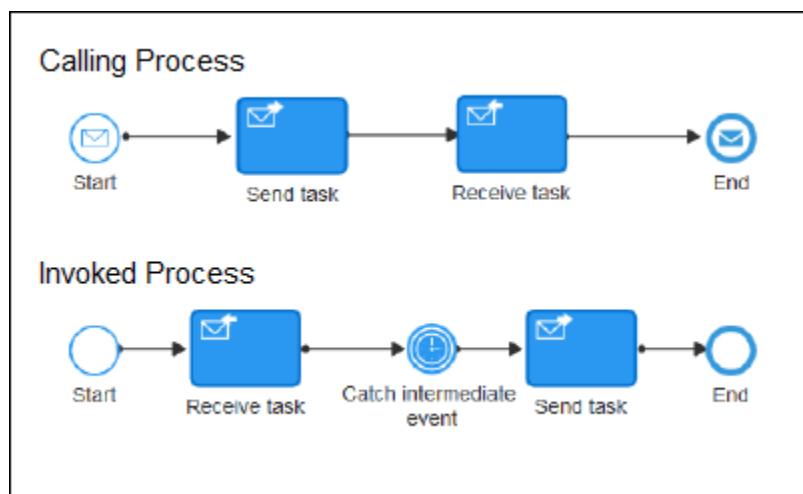
them. The values used to invoke the correlation lead to the identification of the same instance.

To communicate between processes, you can define correlations for the following activities:

- Message start and message end events
- Message throw and message catch events
- Send and receive activities
- Signal events

To define and use correlations in your processes:

1. Go to the Composer page and open your application.
2. Design and build the two processes that will communicate with each other. One process is the **calling process**; the other is the **invoked process**. For example:

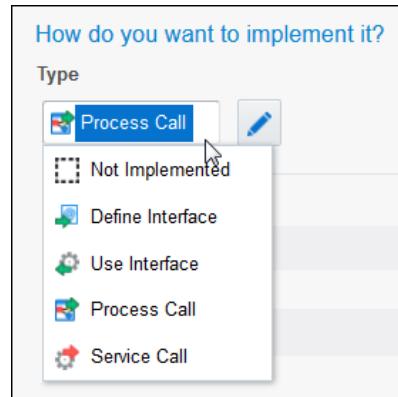


3. Open the properties for an activity.
 - a. Select the activity in your process diagram.

- b. Click **Menu**  and select **Open Properties**.

The Properties pane opens at the bottom of the window.

4. Define the general properties.
 - a. Select the **General** tab.
 - b. Define how you want to implement the activity.



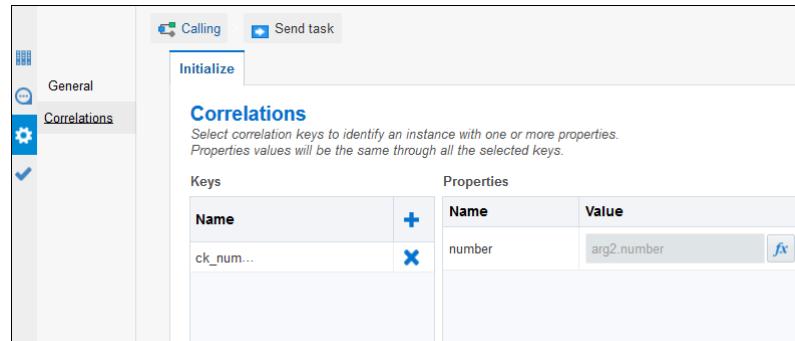
- Click **Edit** to add arguments, and define the name and type of each argument.

Define the input arguments for message start and message catch events and for receive activities. Also, define the output arguments for message end events and for send activities in the invoked process. See [Defining Input or Output Arguments](#).

If the required arguments aren't defined, then you can't configure the correlations. The Correlations tab displays a disabled pane.

- Define the correlation properties.

- Select the **Correlations** tab.



- Define a correlation and configure it to initiate the property values.

A few things to note:

- In the Keys box, click **Add** and select one of the available keys. If no keys are defined, then you need to create them. See [Defining Correlation Keys and Properties](#).
- After you select one of the available keys, the contained properties display in the Properties box. To set values for the properties, click **Expression Editor** to open the editor.
- Some activities are only able to initialize correlations (configured on the Initialize tab) while other activities are able to use the correlations (configured on the Correlate tab). Also, for activities that have input and output arguments, you can select one or both, and then set the property values according to the selected context.

Working with Human Tasks

You can use human tasks to model user interaction with the application. Human tasks enable you to display a form for the user to view or complete, and click an action to perform.

There are different types of human tasks that enable you to model different types of interactions:

- **Submit Tasks:** Enable you to display a form that the user must submit to create a request or to provide information about a certain subject.
- **Approval Tasks:** Enable you to display a form that the user must review or complete and then perform a certain action. The user might approve or reject the request. You can also define custom actions for the user to perform. Approval tasks enable you to define an approval pattern. Generally you use the outcome of the approval task to drive the rest of the process flow.

After you create the human task, you can configure implementation details such as its assignment, priority, due date, reminders and deadlines.

- [Typical Workflow for Creating a Human Task](#)
- [Creating Submit and Approval Tasks](#)
- [Configuring Human Tasks](#)
- [Customizing Notification Emails for Human Tasks](#)

Typical Workflow for Creating a Human Task

Human tasks enable you to model the user interaction with the application. When you add a human task to your process, you must also configure its implementation, define a form, and configure the human task data associations.

When you create a human task you must:

1. Add a human task to an existing business process.
2. Configure the human tasks implementation.
3. Create a form to display the human task information.
4. Configure the human task to use the created form.
5. Configure the human task input and output arguments using data associations.

Creating Submit and Approval Tasks

You can create human tasks to model the user interaction with the application. You can use submit tasks to display a form for the user to complete and submit. You can use approval tasks to display a form for the user to view and/or complete, and then perform an action such as approving it, or rejecting it, or a custom action that you define.

To create a submit task:

1. Edit an existing business process or create a new one.
2. Click the **Show All** button located at the bottom of the Activities toolbar.

3. Select the **Human** task.

The Submit and Approve tasks appear to the left of the toolbar.

4. Click the task that you want to add. Then drag it to the process editor and drop it on the swimlane that represents role that you want to have access to this human task. If you drop the task over a transition, it gets automatically added to the process flow.

The Submit or Approve task is added to the process.

5. Configure the human task by following one of the procedures described in [Configuring Human Tasks](#).
6. Click the **Save and Continue Editing** button located next to the editing label in the toolbar.

A default process data object with the name `TaskOutcomeDataObject` is automatically created for the human task outcome and is shared by all the human tasks in a process. This default data object is automatically associated with the outcome field in the human task output when you click **Save**.

 **Note:**

`TaskOutcomeDataObject` is a reserved name and therefore, you can't use this name when creating data objects.

Configuring Human Tasks

You can configure the assignment of the human task, the form to use to display its information, the title and summary to identify it, its due date and priority, the number of reminders to send to its assignees, and the action to take when it reaches a certain deadline.

To configure a human task:

1. In the process editor, click the human task, select the **Menu** icon, and then select **Open Properties**.

The Properties pane opens at the bottom of the window. The General tab is selected by default.

2. On the **Properties** pane, you can configure the following:

Tab	More Information
General	Working with Draft Processes Assigning Human Tasks Using Forms to Display Task Information Defining an Approval Pattern Configuring the Title and Task Summary Configuring the Due Date and Priority Bypassing the Approval Chain
Reminders	Configuring Reminders
Escalation and Expiration	Configuring Task Expiration, Renewal, or Escalation

Tab	More Information
Notification	Customizing Notification Emails for Human Tasks
Documents	Overriding Documents Folder Access
Conversations	Changing Conversation Settings for a Human Task

3. Save your changes by clicking the **Collapse Pane** icon in the top-right corner of the pane.

Assigning Human Tasks

You can assign the human task to a specific user, to a group of users, to the users in a certain role, or to the same user that already acted on the instance for a certain role. You can also use expressions to calculate the user, group or shared role.

To assign a human task to a specific user:

1. In the General tab on the implementation pane, click **Edit** next to the **Assignee(s)** field.

The Define Assignees dialog box opens.

2. From the **Build a list of participants using** drop-down list, select one of the following:

- Lane Participants
- Names and Expressions

3. If you selected *Lane Participants*, select one of the following options:

- Exclude the following participants:
 - **Creator:** The user who created the process.
 - **Previous Participants:** Users who have already acted within this task instance.
 - **Expression:** User from an expression.

Selecting this option, activates the expression editor icon. After creating an expression, it appears in the text box located next to the check box.

Note:

If you exclude a user from a task, the user will be able to view the task but not have the permission to act on it.

- Any member of current swimlane role: assigns the human task to any participant granted the role required to run the human task.
4. If you selected *Names and Expressions*,
 - a. Exclude the following participants:
 - **Creator:** The user who created the process.
 - **Previous Participants:** Users who have already acted within this task instance.
 - **Expression:** User from an expression.

Selecting this option, activates the expression editor icon. After creating an expression, it appears in the text box located next to the check box.

- b. Click **Add**.
- c. Select one of the available options:
 - Add User
 - Add Group
 - Application Role

A row with the participant type that you selected appears in the List of Assignees table.

- d. Click the cell for the **Data Type** column to select a data type.
To specify the participant using names, select *By Name*.
To specify the participant using an expression, select *By Expression*.
- e. If you selected *By Name*, click **Search** to select a participant.
- f. If you selected *By Expression*, click **Expression** to enter an expression that determines the user or group to assign the human task to.

5. Click **OK**.

The **Define Assignees** dialog box closes and the selected participant or expression appears in the **Assignee(s)** field.

6. Enter the **Percentage Required** and the **Default Outcome**.

These fields only appear if you select *All Assignees in Parallel* option from the **Who are the approvers?** drop-down list. This is the default outcome when the required percentage isn't reached. For example, the required percentage is 51% and the default outcome is *APPROVE*. 55% of assignees vote to reject the task. Therefore, the outcome is *REJECT*.

Using Forms to Display Task Information

You can configure the human task to use a specific form to display the information the user needs to view or complete to perform the task assigned to them. You can use an already existing form, or create a new one.

When you implement a human task with a form, data association is automatically performed when a form is selected:

- If the data object already exists, then that one is used.
- If the data object with the same name doesn't exist, then the first data object of the same type is used.
- If there's no data object of the same type, then a new data object is automatically created. New data objects use this naming convention:

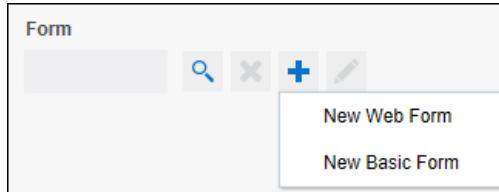
<form-name-starting-in-lower-case>DataObject<n>

where n is a number added to avoid duplicate names.

- Name restrictions for data objects and forms are similar, so no special treatment is required.
- After data association is performed, you're notified with a message below the form input box.

To associate a form with a human task:

1. In the process editor, select a human task, click **Menu**  and select **Open Properties**.
2. In the properties that display, either browse to select a form or create a new one. To create one, click **Add**  next to the **Form** field, and select an option for creating a new form.



3. Enter a form name and click **Create**.

If you select the **Open Immediately** check box, then the form automatically opens in the form editor.

If you're configuring an approval task, you can edit the default actions and add new ones. To add actions, enter their name in uppercase and separate them with commas. You can later access the value of the selected action from the human task data association, using the predefined data object `outcome`.

If you're configuring a submit task, the only allowed action is **SUBMIT**.

Once a form is associated with a human task, you can open the form by selecting the task, clicking **Menu** , and selecting **Open Form**.

See [Working in the Web Forms Editor](#).

 **Note:**

When you copy and paste a human task anywhere within your BPM process, the associated form is not copied.

Defining an Approval Pattern

Approval tasks let you define an approval pattern by specifying actions. By default, the actions **APPROVE** and **REJECT** are already specified. However, you can also define custom actions, such as **HOLD** and **MOREINFO**.

To define an approval pattern:

1. Go to the **Action** field on the General tab in the Properties pane.
The actions **APPROVE** and **REJECT** appear by default.
2. Enter additional custom actions if required.

Custom actions must be in all uppercase letters. Each action must be separated by a comma.

* Action
APPROVE,REJECT,MOREINFO

Configuring the Title and Task Summary

You can specify a title to identify the human task, and a summary to describe it. The title and the summary appear in the user's task list, so that they can easily identify the human task they're looking for without having to view the details. You can specify the title and the summary using plain text, or generate it using expressions.

To configure the title and task summary:

1. Select the human task element in your process diagram and open its Properties pane.
2. Click the **General** tab.
3. Select one of these options for the **Title** and **Task Summary** fields:
 - **Plain Text:** Use a text string to enter the title or summary to identify the human task.
 - **Expression:** Click the **Expression Editor** to specify an expression that calculates the title or summary of the human task. The Expression Editor dialog box opens. See [About Expressions](#).
 - **Reset:** Use this option to clear the value entered in the field and revert to defaults.

 **Note:**

Your plain text string and your expression are maintained so that you can toggle between the two options without losing any data.

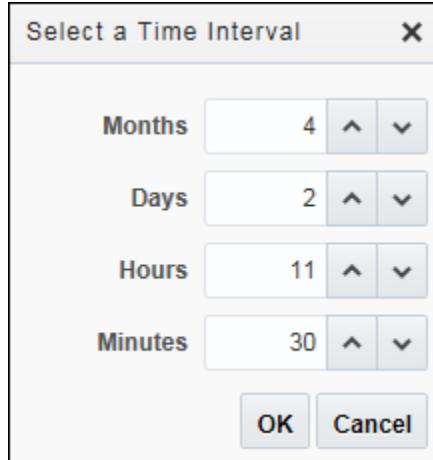
Configuring the Due Date and Priority

You can specify a due date and a priority for a human task. After the due date is reached, the human task is marked as overdue.

To configure the due date and priority:

1. In the General tab in the implementation pane, locate the due date and priority fields.
2. In the **Due Date** field enter an interval to specify the amount of time the assignee has to complete the instance after the human task is triggered. Select one of the following options:
 - **Manually:** Use the format ##M##d##h##m. For example:
 - One hour and thirty minutes: 1h30m
 - One day: 1d
 - Four months, two days, eleven hours and thirty minutes: 4M2d11h30m

- **Expression Editor:** Click the **Expression Editor** to specify an expression to calculate due date of the human task.
The Expression Editor dialog box opens. See [Working with Expressions](#).
- **Interval:** Click **Interval** to specify the due time using the Select a Time Interval dialog box.



- **Reset:** Click **Reset** to clear the value entered in the field and revert to defaults.

 **Note:**

A task due date is different from a process due date. Setting a due date for a process does not automatically set a due date for the task.

3. From the **Priority** drop-down list, select a priority.

Available options are:

- High
- Normal
- Low

These options enable you to sort the Workspace task list based on the task priority. Additionally, in the Workspace task list, high priority tasks are marked with a red exclamation mark.

Bypassing the Approval Chain

You can bypass the approval chain for a specified action on approval type human tasks. For example, if you set the Approvers to be *All Assignees in sequence* and the second of four approvers rejects the task, you can use this feature to bypass the remaining two approvers.

To bypass the approval chain:

1. In the General tab on the properties pane, locate the **Skip Approval on** check box.
2. Select the **Skip Approval on** check box to activate the drop down menu.

* Action <input type="text" value="APPROVE,REJECT,HOLD"/>	<input checked="" type="checkbox"/> Skip Approval on <input type="button" value="APPROVE
REJECT
HOLD"/>
--	--

3. Select the action that you want to skip approval on.

If you add or remove an action in the **Actions** text field, the drop-down menu for skipping approvals automatically updates to reflect the addition or deletion.

Configuring Reminders

You can send reminders to the assignees of a human task. You can specify the number of reminders to send, and the event that triggers the reminder.

To configure reminders:

1. Click the **Reminders** tab on the implementation pane.

The Reminders tab appears.

2. From the **Reminder** drop-down list, select a number of times to send reminders to complete the human tasks to its assignees.

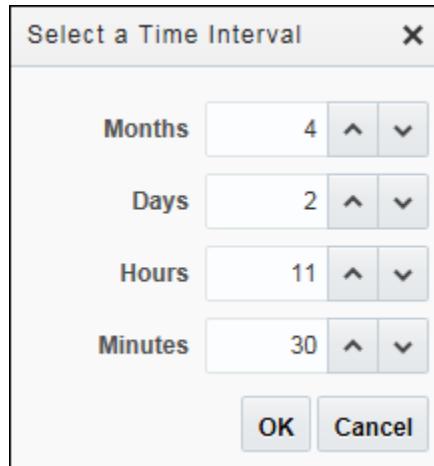
Available options are:

- No Reminder
- Remind Once
- Remind Twice
- Remind Three Times

3. If you selected to send multiple reminders, select an interval to wait between reminders.

You can specify this interval either:

- *Manually*: Use the format ##M##d##h##m. For example:
 - One hour and thirty minutes: 1h30m
 - One day: 1d
 - Four months, two days, eleven hours and thirty minutes: 4M2d11h30m
- *Interval*: Click **Interval** to specify the due time using the Select a Time Interval dialog.



4. If you selected to send reminders, select an event to trigger the first reminder from the **When** drop-down list.

Available options are:

- Before Expiration: send a reminder before the specified expiration time is reached. After the expiration date is reached the human task doesn't appear on the task list.
- After Assignment: send a reminder immediately after assigning the human task to a specific user.
- Before Due Date: send a reminder before the specified due date for the human task is reached. After the due date the human task is marked as overdue, but you can still access it from the task list.

Configuring Task Expiration, Renewal, or Escalation

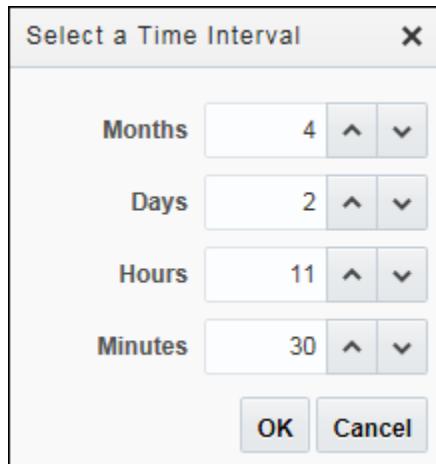
You can configure a human task to never expire, to expire after a certain time, to renew the expiration time, or to escalate after a certain time passes.

To configure an action to perform on a specific deadline:

1. Click the **Escalation and Expiration** tab on the implementation pane. The Escalation and Expiration tab appears.
2. Use the radio buttons to specify if you want the human task to **Never expire**, **Expire**, **Renew**, or **Escalate**.
 - **Never expire**: the human task doesn't expire and if no user completes it, it remains in the users task list for an indeterminate period of time.
 - **Expire**: the human task expires after the specified time and is no longer accessible from the task list.
 - **Renew**: when the specified time passes, the expiration date is extended for one more period until it reaches the specified amount of renewals allowed.
 - **Escalate**: when the specified time passes, the human task is escalated to the specified escalation levels.
3. If you chose the human task to expire, renew, or escalate, specify the interval to wait before performing this action.

You can specify the interval to wait using one of the following methods:

- *Manually:* Use the format ##M##d##h##m. For example:
 - One hour and thirty minutes: 1h30m
 - One day: 1d
 - Four months, two days, eleven hours and thirty minutes: 4M2d11h30m
- *Expression Editor:* Click the **Expression Editor** to specify an expression to calculate due date of the human task.
The Expression Editor dialog box opens. See [Working with Expressions](#).
- *Interval:* Click **Interval** to specify the due time using the Select a Time Interval dialog box.

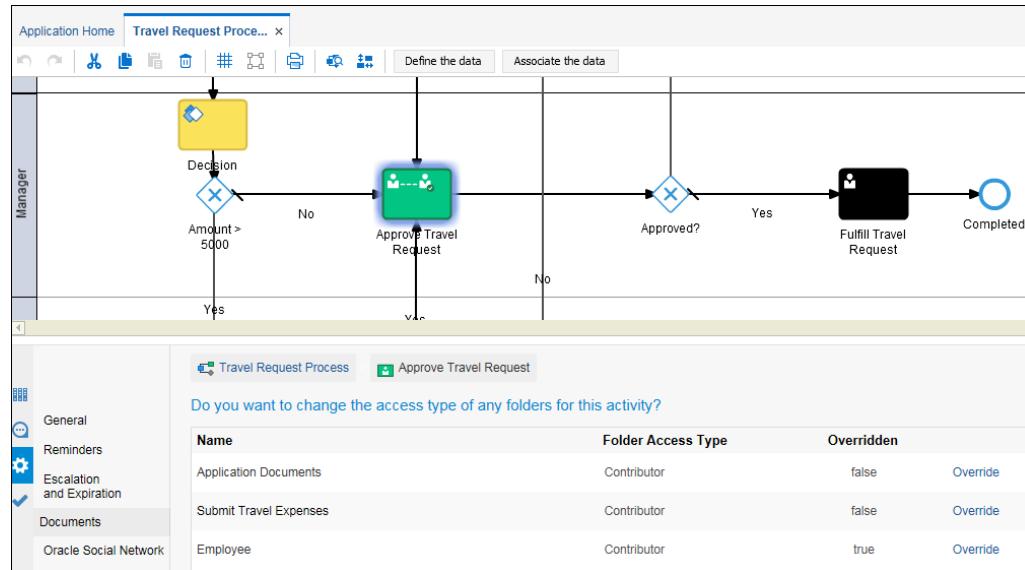


4. If you renew the human task, in the **Maximum Renewals** field, specify the maximum number of times to renew the human task.
5. If you escalate the human task:
 - a. In the **Maximum Escalation Levels**, specify how high in the management chain you want to escalate the human task.
 - b. In the **Highest Escalation Title**, specify to what job title you want to escalate during repeated escalations. For example, *Manager* or *Director*. This is a free-form text field.

Overriding Documents Folder Access

You can override the default folder access for each specific human task in a business process.

The Documents tab of the Implementation pane lists the folders that are available for the human task and allows the user to override the default settings, as shown.



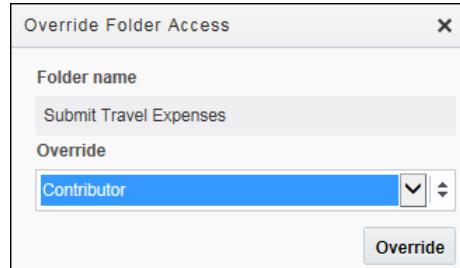
To override access to a documents folder for a specific task:

1. Open the process you want to edit and make sure you're in **Edit** mode if you're editing a shared application.
2. In the process editor, click the human task, select the **Menu** icon, and then select **Open Properties**.

The Properties pane opens at the bottom of the window.

3. Click **Documents** to list the available folders for this task.
4. Click the **Override** link that's located next to the folder you want to change access on.

The Override Folder Access dialog box opens.



5. Select one of the following access types from the drop-down list:
 - Contributor
 - Downloader
 - Viewer
 - None
6. Click **Override** to save your changes.
7. Close the implementation pane with your changes saved by clicking the **Collapse Pane** icon on the top right corner.

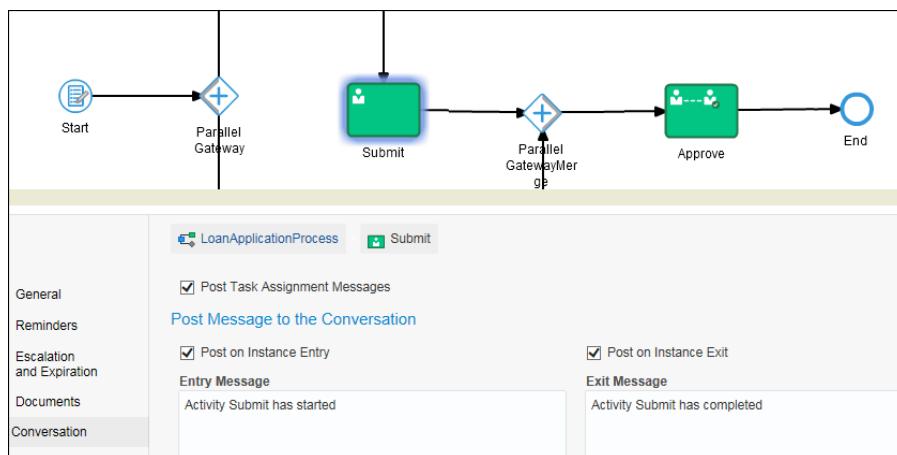
Changing Conversation Settings for a Human Task

The Conversation tab of the Implementation pane enables you to override some of the default conversation property settings for specific human tasks. You can also select whether to post entry and exit messages for specific tasks, and whether to use default messages or create your own.

To change conversation settings for a specific human task:

1. Open the process you want to edit and make sure you're in **Edit** mode if you're editing a shared application.
2. In the process editor, click the human task, select the **Menu** icon, and then select **Open Properties**.

The Properties pane opens at the bottom of the window.



3. Click **Conversation** to display the current conversation settings for this task.
4. Make your changes. You can:
 - a. Select to post task assignment messages
 - b. Select to post messages on instance entry and exit
 - c. Customize entry and exit messages
5. Close the implementation pane with your changes saved by clicking the **Collapse Pane** icon on the top right corner.

Customizing Notification Emails for Human Tasks

Keep your users informed about their task assignments and the progress of a process. You can easily configure notification emails for human tasks and create templates for those notifications.

Topics

- [About Notification Email and Templates](#)
- [Managing Email Templates](#)
- [Configuring Email Templates](#)

About Notification Email and Templates

You can configure Process to send a notification email to the task assignee when an event such as assignment, approval, escalation, reminder, and reassignment occurs. By default, a notification email contains a link that lets the user sign in, view, and then complete the task.

Note:

After you configure the email notifications in design time, you need to enable the email notifications in runtime, see [Enabling Email Notifications](#). You can then view the notification logs and also resend email notifications to all or some of the original recipients, see [Viewing and Resending Email Notifications](#).

You can also configure and use customized notification emails for human tasks. You have an option to create customized email templates that include the following information:

- Payload
- Task
- Links to perform an action on a task
- Task comments

About Actionable Emails

You can configure a notification email with or without actions. An email with actions contains links to perform an action on the task.

WARNING:

Any user who has access to the email can perform the action so be careful who the notification email is sent to.

About Default Email Templates

If you enable email notification for a human task, you can either select a default template or create a new one for your use. By default, the following two email templates are available:

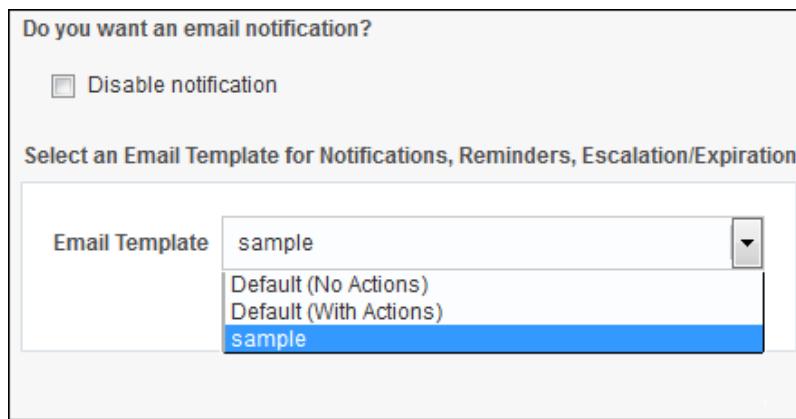
- Default (With Actions) – Contains links to perform an action on the task without having to sign in
- Default (No Actions) – Contains a link that lets the end user sign in to the application, view the task, and complete it. If the end user is viewing the email on a mobile device and clicks the link, then the mobile app launches by default, if it's installed.

Managing Email Templates

Use a customized notification email for human tasks that includes basic task information, form payload values, and comments, and that optionally lets end users perform a task action without having to sign in to the application. Create customized notification emails using the **Notification** options in the Properties pane.

To assign an email template to a human task:

1. Open a process in the process editor.
2. Select a human task, click **Menu** , and select **Open Properties**.
3. Click **Implementation** and select the **Notification** tab.

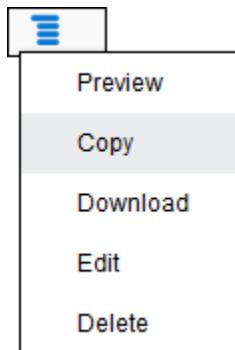


4. If needed, disable or re-enable email notifications. Note that notifications are enabled by default. To disable, select the **Disable notification** check box to stop email notifications during task assignment. Note that disabling the notification for a task won't disable reminder, escalation, or expiration emails. Instead, the selected email template will be used for those events.
5. In the Email Template field, select an email template that you want to use for the task-related emails.

Optionally, create a new email template for your use. Click **Add**  to the right of **Manage Email Templates**. See [Configuring Email Templates](#).

Manage Email Templates	
Name	
Default	
sample	

Optionally, copy, edit, preview, or download an existing email template for your use. Click **Options**  to the right of the email template that you want to use.



Available options include:

Option	Description
Preview	Open the email template in the Preview Email Template dialog. You can view how the email appears to end users. The payload information isn't displayed in the Preview mode.
Copy	Copy and save the email template with another name.
Download	Download and save the email template in HTML format to your local drive or select an application to open the file.
Edit	Open the Edit Email Template dialog. You can edit your template in the code editor or browse for and upload a template from your local drive. You can also preview the template before finalizing it. This option isn't available for the default template.
Delete	Delete the email template. This option isn't available for the default template. Important: Deleting a template that is currently in use deletes all references to the template.

Configuring Email Templates

Create customized email templates and use them to send notification emails for human tasks. Upload email templates from your local drive, edit or copy existing email templates, or download email templates for your use.

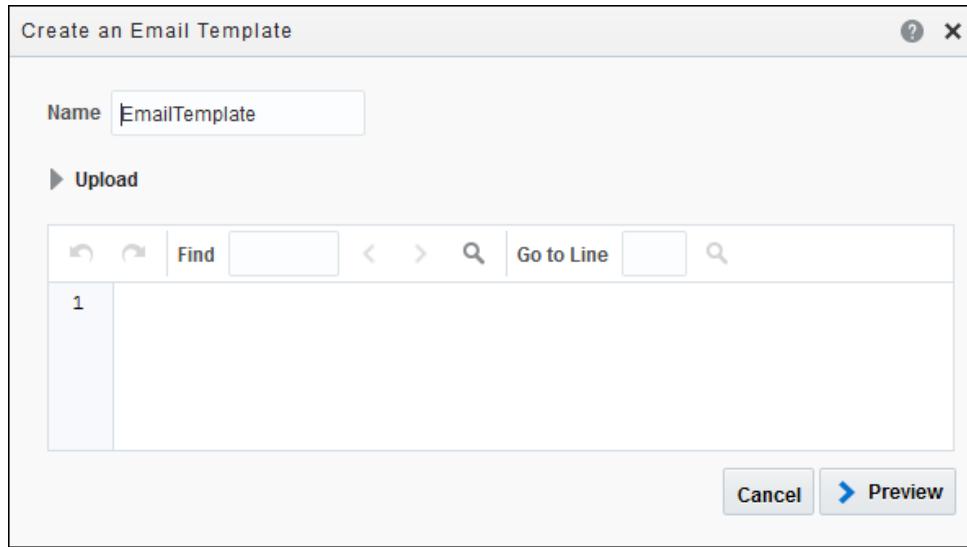
Creating an Email Template

You can create email templates by uploading a file with an HTM or HTML extension, or manually entering the HTML markup directly into the code editor provided in the Create an Email Template dialog box.

To create a new email template:

1. Open a process in the process editor.
2. Select a human task, click **Menu** , and select **Open Properties**.
3. Click **Implementation** and select the **Notification** tab.
4. Click **Add +** next to Manage Email Templates.

The Create an Email Template dialog box opens.



5. Enter a name for your email template.
6. Use one of these options to create your email template:
 - You can click **Upload** to browse for and add an existing email template. You can then edit its contents.
 - Alternatively, you can start from scratch by manually entering the HTML markup for your email template. To include data, you'll use mustache templates. Mustache templates are logic-less and work by expanding tags in a template using values provided in an object—the task object in this case. These templates provide an easy way to include data references in an HTML file. The following example shows how to reference the first name and last name using the mustache template:

Template:

```
<p>Hello {{first_name}} {{last_name}}</p>
```

Object:

```
{
  "first_name": "Joe",
  "last_name": "Smith"
}
```

Output:

```
<p>Hello Joe Smith</p>
```

See <https://mustache.github.io/> and <https://mustache.github.io/mustache.5.html>.

The following table provides a description of the **task** related attributes of the task object. Note that the variable names are case-sensitive.

Variable	Description
acquiredBy	Name of the user who has acquired the task
acquiredById	ID of the user who has acquired the task
assignedDate	Date when the task was assigned

Variable	Description
createdDate	Date of creation of the task
creator	Creator of the task
creatorId	ID of the creator of the task
dueDate	Due date for completing the task
endDate	Date when the task must end
fromUser	Creator or user who reassigned the task
fromUserId	ID of the user from whom the task was acquired
longSummary	Detailed description of the task
outcome	Task outcome
ownerGroup	Group to which the task owner belongs
ownerRole	Role of the task owner
ownerUser	User who owns the task
ownerGroupId	ID of the group to which the task owner belongs
ownerRoleId	ID of the role of the task owner
ownerUserId	ID of the user who owns the task
priority	Task priority
payload	Map of task payload
priorityNum	Priority number of the task
shortSummary	Short summary of the task
startDate	Start date of the task
state	Current status of the task
taskId	Task ID
taskNumber	Task number
taskDefinitionId	Definition ID of the task
title	Title of the task
updatedBy	Name of the user who has updated the task
updatedById	ID of the user who has updated the task
updatedDate	Date when the task was updated

The following table provides a description of the **process** related attributes of the task object:

Variable	Description
instanceId	ID of the process instance in which the task is present
processId	ID of the process in which the task is present
processName	Name of the process in which the task is present
processVersion	Version of the process in which the task is present

The following table provides a description of the **application** related attributes of the task object:

Variable	Description
assignee	Full name of the task assignee
currentYear	Current year
logo	Oracle logo
url	URL for accessing the task details in runtime

The following table provides a description of the **comments** related attributes of the task object:

Variable	Description
comments	List of comments
commentStr	Actual comment
updatedBy	Name of the user who updated the comment
updatedDate	Date the comment was updated

The following table provides a description of the **action** related attributes of the task object:

Variable	Description
actionDisplayName	Display name of the action. For example, Approve, Reject.
actionName	Name of the action defined in human task action. For example, APPROVE, REJECT. Note that the action name is case sensitive and must match the task outcome.
actions	List of actions.
url	URL for performing the action.

Examples:

The following table provides examples of how to define tags using mustache templates:

To define a...	Example
Variable	<code>{{title}}, {{assignee}}</code>
Section/List	<code>{{#comments}} {{commentStr}} {{updatedBy}} at {{updatedDate}} {{/comments}}</code>
Null check	<code>{{#dueDate}} Due Date: {{dueDate}} {{/dueDate}}</code>
HTML string	<code>{{{actions}}}</code>

To define a...	Example
Payload	<p>Use the camelCase naming convention for all business objects and form keys used in the payload to get the desired email template.</p> <p>Payload Example for a Web Form with Business Object</p> <pre> {{#payload}} {{#financialApprovalForm}} <td valign="top">{{contractCategory}}</td> <td valign="top">{{creditLimit}}</td> {{#contractDataObject}} <td valign="top">{{customerName}}</td> <td valign="top">{{contractStartDate}}</td> <td valign="top">{{creditLimit}}</td> {{/contractDataObject}} {{/financialApprovalForm}} {{/payload}} </pre>

You can view the structure of payload in the Data pane on the web form page.

The screenshot shows the Oracle HCM Cloud Data pane with a tree view of the payload structure:

- NAME**: financialApprovalForm
- TYPE**: (FinancialApprovalForm)
 - contractCategory**: (string)
 - contractDataObject**: (ContractDataObject)
 - customerName**: (string)
 - contractStartDate**: (dateTime)
 - contractEndDate**: (dateTime)
 - creditLimit**: (double)
 - approvalNotes**: (string)
 - region**: (string)
 - creditLimit**: (double)

Payload Example for a Basic Form with Text and Boolean Data

```

{{#payload}}
  {{#travelRequestForm}}
    {{#form}}
      <td valign="top">{{name}}</td>
      <td valign="top">{{customerName}}</td>
    {{/form}}
    <td>
      <td valign="top">{{totalAmount}}</td>
      <td valign="top">{{country}}</td>
      <td valign="top">{{city}}</td>
      <td valign="top">{{purposeOfVisit}}</td>
    {{/form}}
  {{/travelRequestForm}}
{{/payload}}

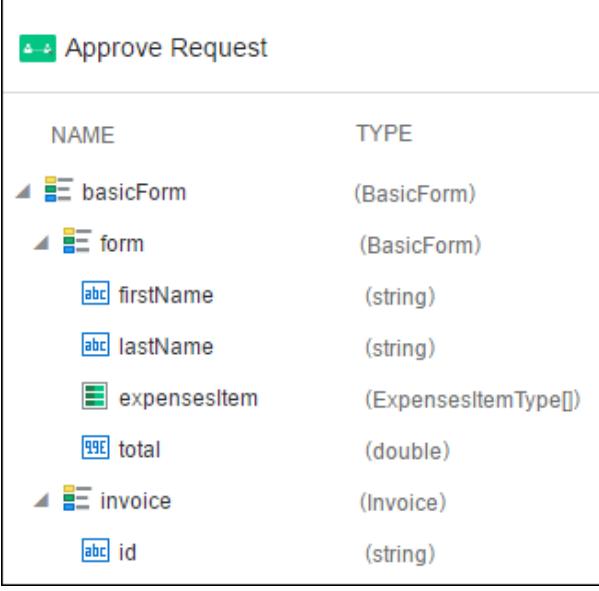
```

To define a...	Example
To get the structure of the payload, see the incoming webform dataObject in the Data Association page of the human task.	<pre> graph TD travelRequestForm[travelRequestForm (TravelRequestForm)] --> form[form (TravelRequestForm)] form --> name[name (string)] form --> date[date (date)] form --> totalAmount[totalAmount (string)] form --> country[country (CountryType)] country --> USA[USA (CountryType)] country --> INDIA[INDIA (CountryType)] form --> city[city (CityType)] city --> OPTION_3[OPTION_3 (CityType)] city --> OPTION_1[OPTION_1 (CityType)] city --> OPTION_2[OPTION_2 (CityType)] form --> purposeOfVisit[purposeOfVisit (PurposeOfVisitType)] purposeOfVisit --> INTERNAL[INTERNAL (PurposeOfVisitType)] purposeOfVisit --> CUSTOMER[CUSTOMER (PurposeOfVisitType)] purposeOfVisit --> TRAINING[TRAINING (PurposeOfVisitType)] purposeOfVisit --> SEMINAR[SEMINAR (PurposeOfVisitType)] form --> customerName[customerName (string)] form --> justification[justification (string)] form --> amount[amount (double)] </pre>

Payload Example for Basic Form with Array Data and Business Object

```

{{#payload}}
{{#basicForm}}
{{#form}}
{{#expensesItem}}
<td valign="top">{{firstName}}</td>
<td valign="top">{{lastName}}</td>
{{#expensesItem}}
<td
valign="top">{{description}}</td>
<td valign="top">{{amount}}</
td>
{{/expensesItem}}
{{/form}}
{{#invoice}}
  
```

To define a...	Example
	<pre><td valign="top">{{id}}</td> {{/invoice}} {{/basicForm}} {{/payload}}</pre>  <p>The screenshot shows a tree structure of variables under the node 'Approve Request'. The variables and their types are:</p> <ul style="list-style-type: none"> basicForm (BasicForm) form (BasicForm) <ul style="list-style-type: none"> firstName (string) lastName (string) expensesItem (ExpensesItemType[]) total (double) invoice (Invoice) <ul style="list-style-type: none"> id (string)
Actions	<p>You can define actions using a string, a list, or a map.</p> <ul style="list-style-type: none"> Actions as a string: In this case, actions are defined as per the default email. You can't modify the look and feel of the links. For example: {{{actions}}} Actions as a list: Use it if you want to iterate over each action to modify the look and feel of links. The same setting is applied on all the links. For example: <pre>{{{#actionsList}}} {{actionDisplayName}} {{/actionsList}}</pre> <ul style="list-style-type: none"> Actions as a map: Use it if you want to customize each action link. For example: <pre>{{{#actionsMap}}} {{{#APPROVE}}} {{actionDisplayName}} {{/APPROVE}} {{{#REJECT}}} {{actionDisplayName}} {{/REJECT}} {{/actionsMap}}</pre>

- Click **Preview** to view your template before finalizing it.

The screenshot shows a web-based application for managing tasks in the Oracle Cloud. At the top, it says "ORACLE® Cloud". Below that, a message to the assignee is displayed: "Hello Assignee," followed by "Task Title of the task requires your attention . [View Online](#)". A "Task Information" section contains fields for "From", "Summary", "Priority", "Created On", and "Due Date". Below this is a "Data" section which is currently empty. Under "Comments", there are two entries: "Comment String 1" (By - User1 at Update Date) and "Comment String 2" (By - User2 at Update Date). At the bottom, there is a copyright notice: "Copyright 2016. Oracle and/or its affiliates. All rights reserved." followed by links to "About Oracle | Legal Notices and Terms of Use | Privacy Statement". A note below states: "This is a system generated message. Do not reply to this message. You are receiving this email as a result of your current relationship with Oracle Cloud. General marketing opt-out preferences have been over-ridden to ensure that you receive this email." At the bottom right are buttons for "Cancel", "Previous", and "Finish".

 **Note:**

In preview mode, the payload and action map sections don't show the data.

The actual email looks something like this:

Hello Amy Silver ,

Task New Hire Approval requires your attention . [View Online](#)

APPROVE HOLD REJECT

Task Information

From: James Cooper
Summary: Go through the candidate profile to make a decision on hiring
Priority: High
Created On: 2017-02-02 at 08:33:56
Due Date: 2017-02-05 at 08:33:56

Data

Candidate information

Name: John Smith
Department: Computer Science

Education

University: State University
Start Date: 2010-08-05T07:00:00Z
End Date: 2014-05-23T07:00:00Z

Comments

Copyright 2016. Oracle and/or its affiliates. All rights reserved. [About Oracle](#) | [Legal Notices and Terms of Use](#) | [Privacy Statement](#)
This is a system generated message. Do not reply to this message. You are receiving this email as a result of your current relationship with Oracle Cloud. General marketing opt-out preferences have been over-ridden to ensure that you receive this email.

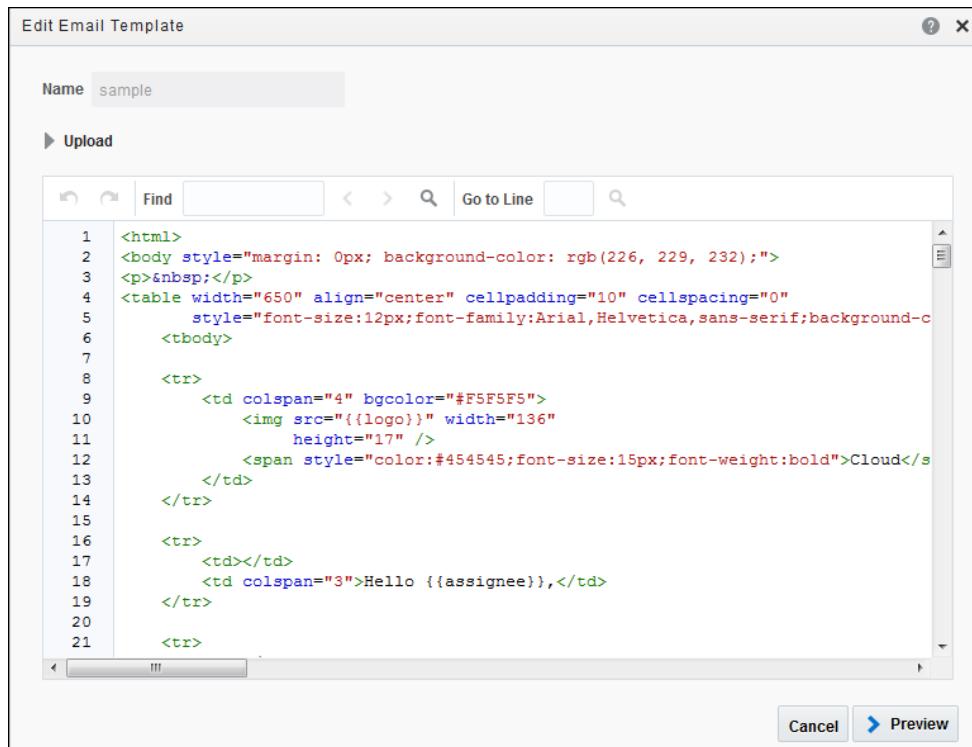
Editing an Existing Template

You can edit an existing template using the Edit Email Template dialog box.

To edit an existing email template:

1. Open a process in the process editor.
2. Select a human task, click **Menu** , and select **Open Properties**.
3. Click **Implementation** and select the **Notification** tab.
4. Click **Options**  next to the email template that you want to edit.

The Edit Email Template dialog box opens.



5. Edit the HTML markup as required.

Alternatively, click **Upload** to browse for and upload an existing email template to replace the current one.

6. Click **Preview** to view your template before finalizing it.

Creating Web Forms

In Process, you create web forms to interact with end users. As part of creating a web form, add its controls, configure its data, and define form behavior.

Note:

Oracle Process Cloud Service provides two form editors: the **web forms editor** (recommended, described below), and the **basic forms designer** described in [Creating Basic Forms](#).

Topics

- [Working in the Web Forms Editor](#)
- [Ready to create a web form?](#)
- [Working with Presentations](#)
- [Positioning Controls on Forms](#)
- [Binding Form Data with Controls](#)
- [Working with Stylesheets and Styling](#)
- [Configuring Basic Controls](#)
- [Configuring Advanced Controls](#)
- [Creating Computed Controls](#)
- [Localizing Web Forms](#)
- [Previewing Forms and Their Payload](#)
- [Adding Dynamic Behaviors to a Form](#)
- [Reusing Forms](#)

Working in the Web Forms Editor

Use the web forms editor to create forms for your human tasks and start form events without scripting rules.

If you're new to the editor, [start here to learn the basics](#) by creating and configuring a web form and then trying it out in runtime.

Want to learn more about configuring a specific control? See [Basic Controls](#) or [Advanced Controls](#).

Accessing the Web Forms Editor

Creating or opening a web form displays the editor. Add, edit, open, or delete web forms using either of these methods:

- **On the Forms page**

On the Application Home tab, select **Forms** in the Components pane. The Forms page displays all forms defined for the application, including web forms and [basic forms](#).

Add, edit, or delete a web form on this page. Note that to activate a web form, you must associate it with a flow element, by selecting it in a [human task](#) or [form start event](#) in a process.

- **From the process editor**

On the Application Home tab, select **Processes** in the Components pane and open a process. Select a human task or start form event and open its properties. Add, edit, select, or delete a form on the human task's implementation pane. Adding or selecting a web form in this way associates it with the selected human task or start form event.

Using the Web Forms Editor

In the editor, drag, drop, and [position controls](#) onto the form's central canvas. Select from a wide variety of controls on the [Basic](#) and [Advanced](#) Palettes, ranging from standard text and drop-down fields to tables, list of value (LOV) fields, sections, panels, and special fields such as URLs, videos, and images. Alternatively, [automatically create a form from a business type](#) defined for the application.

After adding some controls, further develop your form by [creating multiple views \(presentations\)](#), [reusing other forms in the form](#), and customizing its styling by [uploading a CSS stylesheet](#) and [applying styling properties](#).

Working with Web Form Data

An integral part of implementing a web form is defining and using its data, both within the form itself and in the human task or start form event.

- Within a web form, you bind form controls to data attributes that hold input values entered by end users. You can [bind data to controls](#) automatically or manually. You can also [automatically create web forms based on business types](#) defined for the application.
- Within a process, you will likely make use of web form data through [data association](#), which defines input and output for flow elements that need them. Process creates data associations for you when you create human tasks with a web form or form start events. See [Configuring Data Association](#).

Building Dynamic Forms Using Events and REST Connections

Incorporate dynamic behavior in web forms without scripting by [configuring events on controls](#) that upon firing, trigger conditions, actions, or REST connector calls. Populate controls [from REST or data connections](#).

Associating Forms with Flow Elements

If you've created a form but haven't associated it with a process, select **Processes** in the Components pane of the Application Home tab and open a process. Select a human task or start form event and open its properties. In the implementation pane, click **Browse** and select a form from those defined for the application. Note that you can select a web form or a basic form. You can use either or both types of forms in processes. If associating a web form with a flow element, also select a [presentation](#) to display. (Basic forms don't use presentations.)

Ready to create a web form?

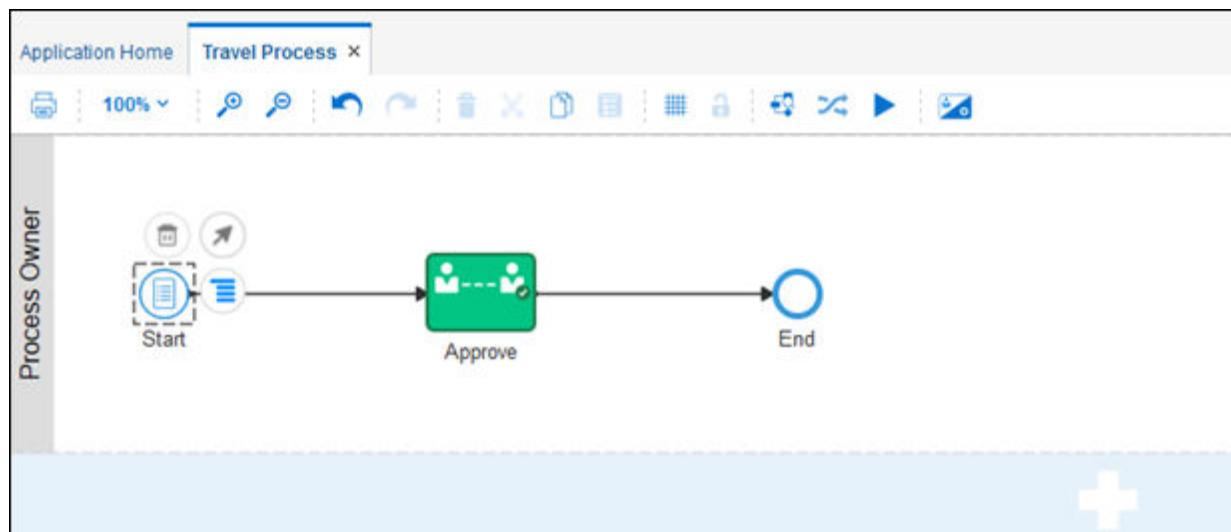
If so, we'll take you through the main steps—from creating a web form to testing it as an end user. You'll learn about forms and their controls, pick up web forms editor tips, and complete the entire web form development life cycle.

- [Create a Simple Application](#)
- [Create a Web Form](#)
- [Add and Configure Controls](#)
- [Add Another Presentation](#)
- [Change the Form's Stylesheet](#)
- [Preview the Form](#)
- [Use Forms and Presentations in a Process](#)
- [Define a Control's Behavior](#)
- [Try the Form in Runtime](#)
- [Explore Advanced Form Options](#)

Create a Simple Application

Web forms enable humans to interact with a business process. For example, a form can start an application or be used in a human task. Let's start by creating a simple application that contains those elements for use in a travel request form.

1. In Composer, click **Create**, then **New Application**.
2. In the Create Application dialog box, enter **Travel**, and click **Create**.
3. In the Create a Process page, select **Start with a form**.
4. In the Create Process window, enter **Travel Process**, and click **Create**.
5. Add an Approve human task to the process and title it **Approve**.

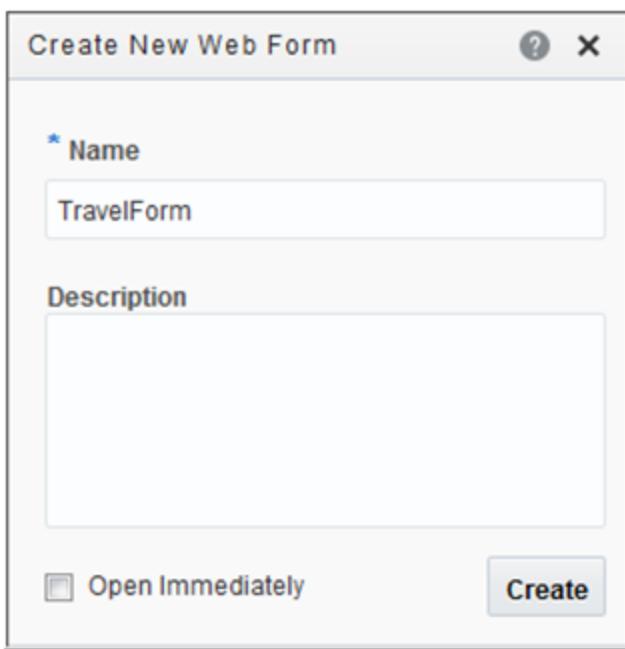


See [Creating Your First Application \(a Quick Start\)](#).

Create a Web Form

Let's create a form for employees to complete with travel request details. This form should also display to users responsible for approving or rejecting the request.

1. In the process, open the Start Form's properties. Click the Start icon, and click its actions menu. (Notice how the **Open Form** command is dimmed, because no form is associated with the flow element yet.) Select **Open Properties**.
2. In the **Title** field, enter **Request Travel**.
3. Click **Create New Form**  by the **Form** field, and select **New Web Form**.
4. In the Create New Web Form dialog box, enter **TravelForm** in the **Name** field, and click **Create**.

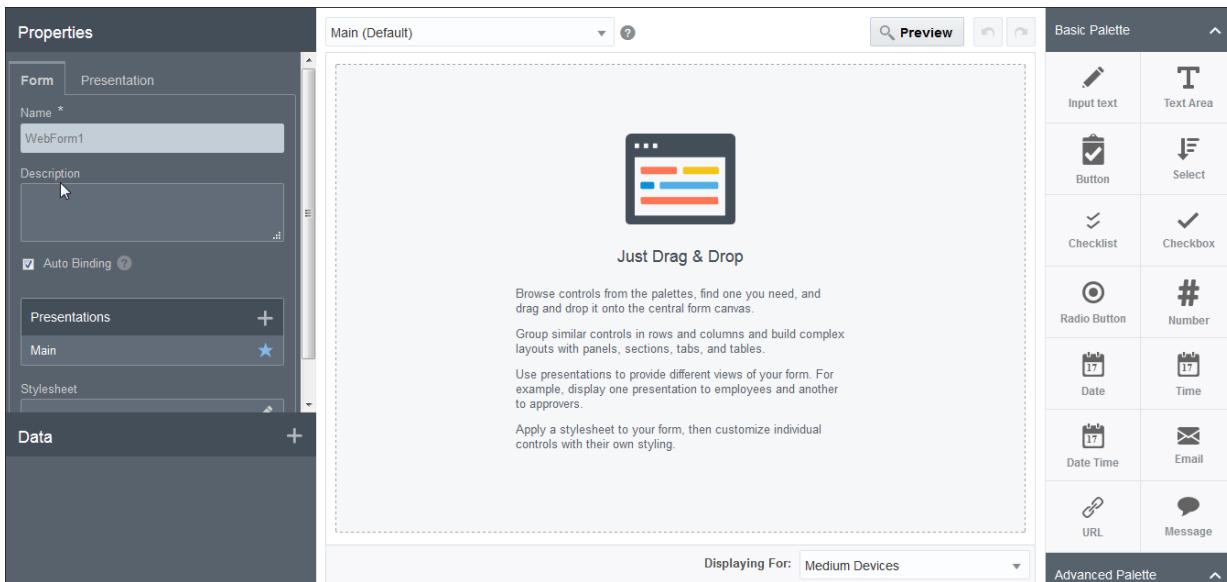


In the Properties pane, notice how the **Form** field identifies the new form.

5. Click the **Edit Form** icon.

The web forms editor opens in a new tab named with the title you entered for the form.

The editor contains a central canvas on which you construct the form by dragging and dropping available controls from the palettes onto the canvas. The **Properties** pane lets you select settings. The **Data** pane lists data attributes for the form's controls.



Add and Configure Controls

Let's add controls from the basic palette and the advanced palette to the travel request form.

1. Add and configure name fields onto the form's canvas in a row.
 - a. From the Basic Palette, drag and drop an Input Text control onto the canvas.
 - b. Drag and drop another Input Text control next to the first control.

As you drag, a box with a dotted outline shows where you can place the control, such as next to, between, or below another control. When you drop, controls are adjusted on an invisible grid to make space. (You can place up to 12 controls in a row across the form.)

To reposition a control, drag its dotted handle and drop it in a new location. The controls around it adjust accordingly. As you edit, click **Undo** and **Redo** as needed.

 - c. Select the first text control. Notice how the Properties tabs are **General** and **Styling**. The tab settings apply to the selected control.
 - d. On the General tab, change the **Name** field to `FirstName`, and the **Label** field to `First Name`. (The **Name** field applies to the control itself and the **Label** field determines its display name.)
 - e. Repeat steps c and d to change the second control's name to `LastName` and its label to `Last Name`.
2. Add and configure date fields.
 - a. Drag and drop two Date controls in a row below the name controls.
 - b. Select the first date control and change its name to `StartDate` and its label to `Start Date`. Scroll down on the **General** tab and select a date format in the **Format** field (for example, `MM/dd/yy`).
 - c. Select the second date control, change its name to `EndDate` and its label to `End Date`, and set its format in the **Format** field.

3. Add a travel justification text control.
 - a. Drag and drop a Text Area control below the date controls.
 - b. Select the control and change its name and label to **Justification**.
4. Add and configure a cost control.
 - a. From the Advanced Palette, drag and drop a Money control above the justification text control.
 - b. Select the control, change its name and label to **Cost**, and select your currency in the **Currency** field.
5. Add and configure an expenses control.
 - a. Position a Repeatable Section control below the justification text control.
 - b. Select the control and change its name to **Expenses**. Scroll down on the **General** tab and select **Users can Add/Remove Rows**.
6. Add and configure an expense details control.
 - a. Drag and drop a Table control on to the expenses section.
 - b. Select the control, change its name to **ExpenseDetails** and its label to **Expense Details**.
 - c. Scroll down on the **General** tab to the Columns section and click **Add**  to add another column. Rename **Column** to **Expense Type** and **Column 1** to **Amount**.
 - d. Scroll down further and select **Users can Add/Remove Rows**.
7. Configure the expense type column.
 - a. Position a Select control into the dotted line area within the expense type column.
 - b. Select the control and change its name to **SelectExpenseType** and its label to **Select Expense Type**.
 - c. Scroll down on the General tab and enter **Select expense type...** in the **Placeholder** field.
 - d. Scroll down further to the **Options Source** section, and enter the following for the **Options Name** and **Options Value** fields:
 - Airfare
 - Cab
 - Hotel
 - Restaurant
 - Others

Options Source ?

Static From Data Connector

Options Names	Options Values
Airfare	Airfare
Cab	Cab
Hotel	Hotel
Restaurant	Restaurant
Others	Others

Note:

The options name and the options value count must be the same.

8. Configure the amount column.
 - a. Drag and drop a Money control into the dotted line area within the amount column.
 - b. Select the control, change its name to `AmountSpent`, and its label to `Amount Spent`, and select your currency in the **Currency** field.

Name * ?
AmountSpent

Label ?
Amount Spent

Binding ?
amountspent

Computed Value ?

Currency ?
US Dollar (USD)

Trip Details x Expense Details x Add tab

RepeatableSection

Expenses

Expense Type	Amount
Select an Expense Type	\$ 0.00

Drop Elements to this panel

9. Switch between control and form properties.

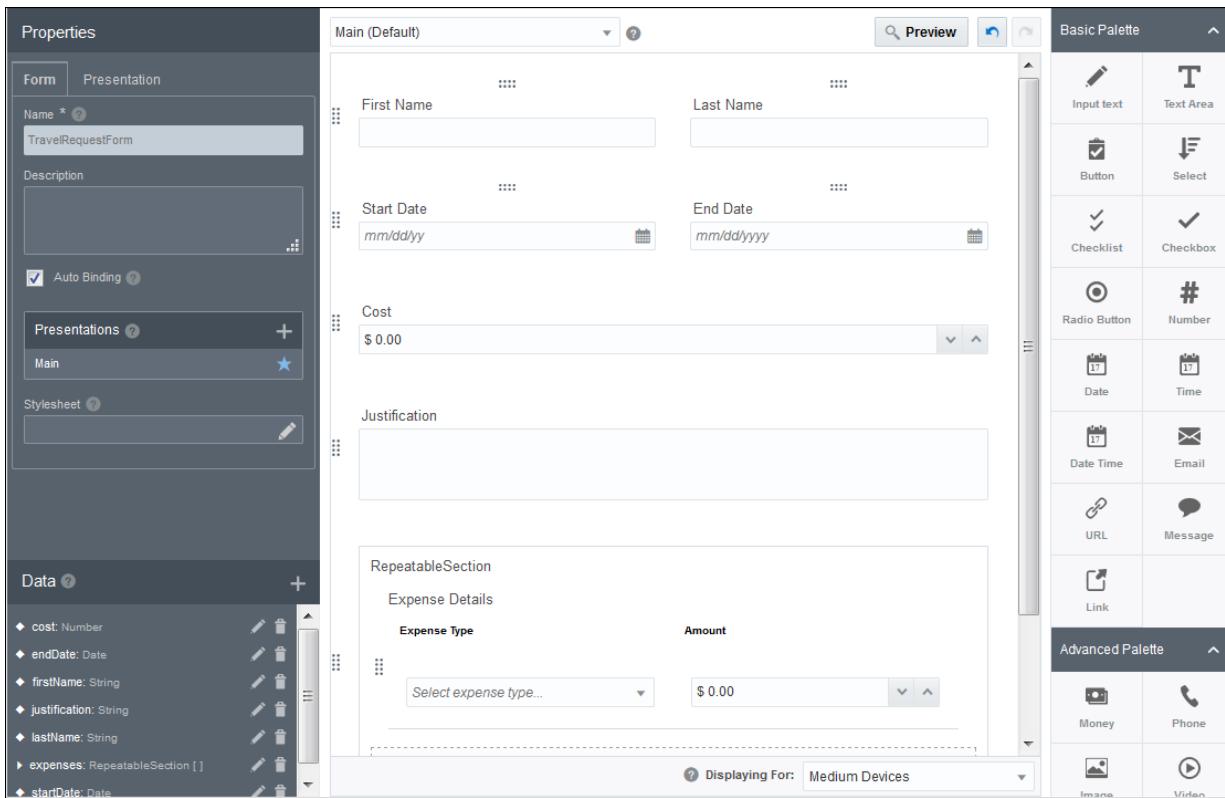
- a. Click a blank area of the form canvas.

Notice that the Properties tabs changed to **Form** and **Presentation**, and now apply to the entire form rather than the selected control. Click a control and the tabs change to **General** and **Styling**, and apply to the control.

Also notice that the Data pane lists data attributes with the same names as your controls (but different capitalization). These attributes were automatically created as you added each control. That happens when you have the **Auto**

Binding field on the **Form** tab selected. These attributes hold the form's payload (working) data while the process is running.

- b. Click **Save** to save the form.



Add Another Presentation

Let's add an alternate view of the form to display to approvers who review travel requests.

1. Click the **Presentation** tab in the **Properties** pane. (If you see **General** and **Styling** tabs instead, select a blank area of the canvas first to display the **Presentation** tab.)
2. In the **Name** field, replace **Main** (the default name) with **Employees**. This is the default presentation that was already created.
3. Click the **Form** tab. Notice that the Presentations table now lists our presentation as **Employees**. The star indicates that it's the default presentation.
4. In the Presentations table, click **Add** to add a presentation. In the Create Presentation dialog box, enter **Approvers** in the **Name** field, select **Employees** in the **Based on** field, and click **Create**.

The screenshot shows a 'Create Presentation' dialog box. The 'Name' field contains 'Approvers'. The 'Based on' dropdown is set to 'Employees'. There are two checkboxes at the bottom: 'Make default' (unchecked) and 'Switch to this presentation' (checked). At the bottom right are 'Create' and 'Cancel' buttons.

5. Use the **Presentation** field at the top of the canvas to switch between your two presentations.

Now let's make the two presentations different.

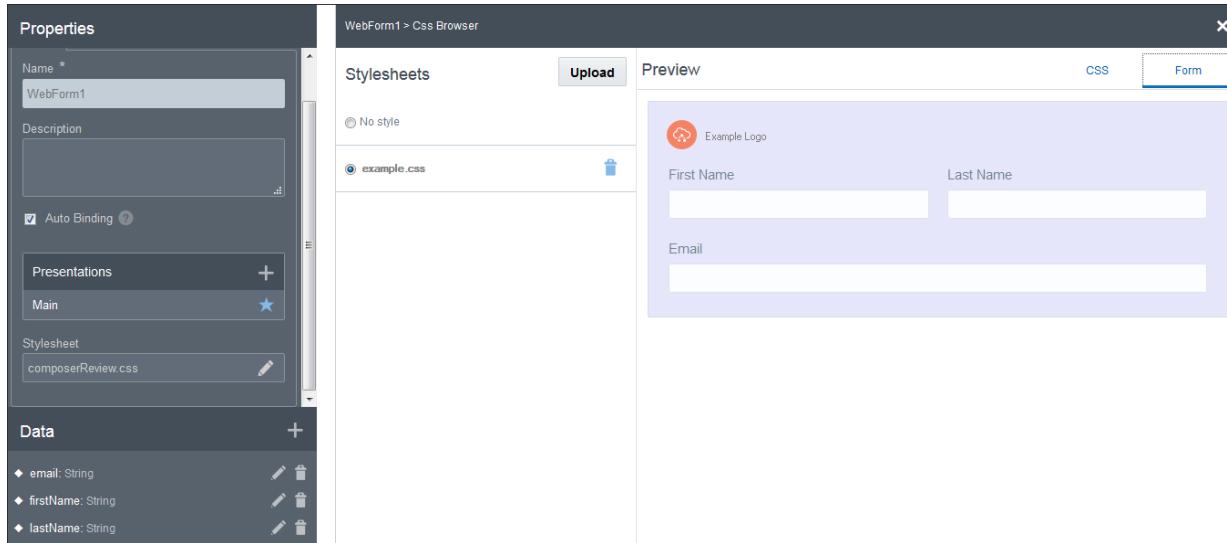
6. Select different background colors for the presentations.
 - a. Select the Approvers presentation, and click the **Presentation** tab.
 - b. Click the square below **Background Color**, select a light yellow color, add it to custom colors, and click **OK**. Optionally select a border color.
 - c. Repeat the same steps to change the Employees presentation's background color to light green.
7. Make the name fields on the Approvers presentation read only.
 - a. Select the Approvers presentation.
 - b. Select the **First Name** field. On the **General** tab, scroll down and select **Read Only**. The field turns blank to indicate that its value can't be changed.
 - c. Change the **Last Name** field to read only.

Change the Form's Stylesheet

To apply your organization's branding, you can upload and apply a stylesheet. You can assign one stylesheet to a form.

1. Click a blank area of the form, then click the **Form** tab on the Properties pane.
2. Scroll down if needed, and click **Edit** in the **Stylesheet** field. The Stylesheets page displays.

3. To upload a stylesheet, click **Upload** and select a stylesheet file (.css or .txt). In the **Stylesheets** area, click an available stylesheet to apply it.
4. In the **Preview** area, click **CSS** to view a stylesheet's format or **Form** to preview it.
5. Click the **X** to close the CSS browser and return to the form.



Preview the Form

Forms you create automatically adapt to the screen size, whether end users display them on mobile devices, tablets, or on large screens.

1. In the web forms editor, click **Preview**.
2. In the Preview window, select different device sizes in which to view the form.
3. Enter sample values in the form's controls and click **Submit**.

The field at the bottom of the window shows how form entries (also called *payload values*) are stored as data attributes and used in the process.

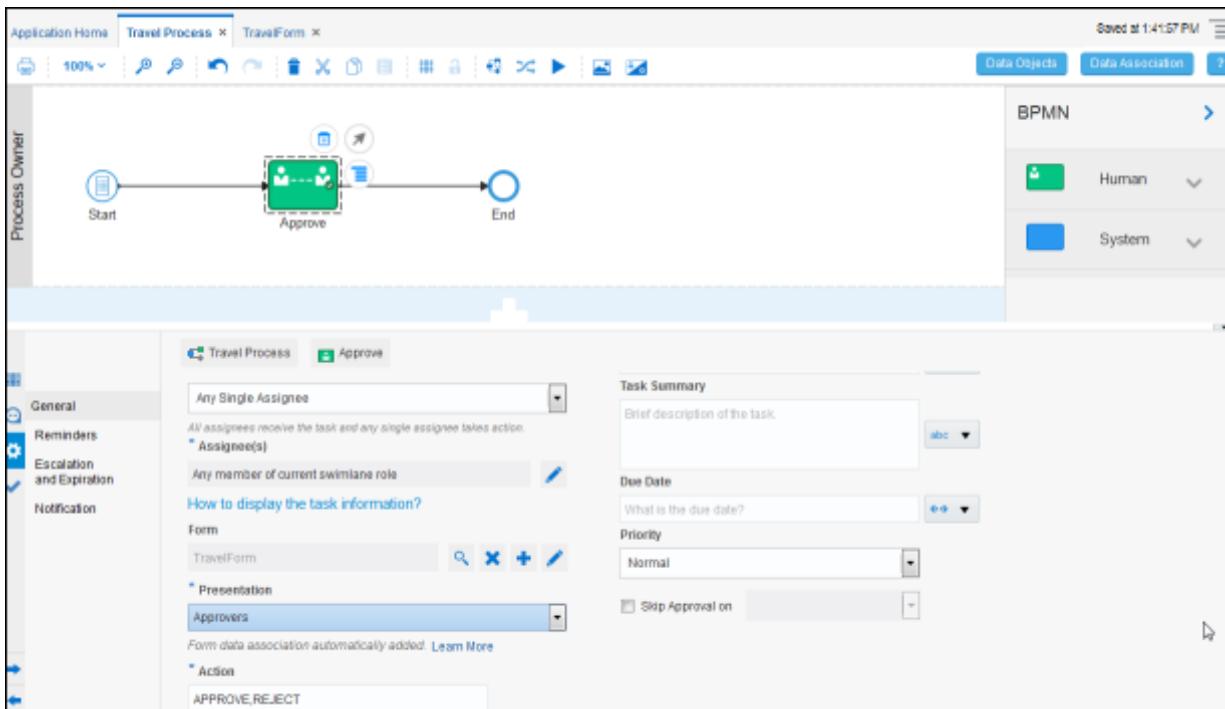
4. Click **Close** to close the Preview window. Click **Save**.

The screenshot shows a web-based form builder interface. At the top, there are tabs for 'Small Devices', 'Medium Devices' (which is selected and highlighted in blue), 'Large Devices', and 'Extra Large Devices'. To the right of the tabs is a 'Fit Screen' button. The main area displays a form for a travel expense. The form includes fields for First Name ('Charles'), Last Name ('Dickens'), Start Date ('05/11/17'), End Date ('05/18/2017'), Cost ('\$ 550.00'), and Justification ('Customer Visit'). Below these fields is a section titled 'Expense Details' containing a table with two rows: 'Expense Type' ('Airfare') and 'Amount' ('\$ 550.00'). There are also '+' and '-' buttons for adding or removing rows. In the bottom right corner of the form area is a 'Submit' button.

Use Forms and Presentations in a Process

Now let's use the Employees presentation in the start form, and the Approvers presentation in the Approve task.

1. Click the **Application Home** tab, then **Forms**. Notice that your form displays here. The Forms pane displays all forms created for the application, regardless of whether they're used. Note that you can create and use both web and basic type forms in an application.
2. Click the **Travel Process** tab.
3. Select the Approve human task, and click its action menu. Notice that the **Open Form** command is dimmed, because no form is associated with it yet. Select **Open Properties**. The pane expands to display its properties.
4. In the **Form** field, browse and select **TravelForm**.
5. In the **Presentation** field, select the **Approvers** presentation to display for this task.



Define a Control's Behavior

To change a control's behavior, make use of the web form editor's extensive set of event, action, condition, and connector options. For your travel request, let's apply an event, condition, and action to the Cost control: if the cost is over a certain amount (such as \$500), the Justification text field can't be left blank.

- From the Approve task's properties, click the **Edit Form** icon for the **Form** field.
- In the web forms editor canvas, select the Cost control. On the **General** tab, scroll down to the **Events** field and click **Add** to create an event.

An **event** represents something that may occur, which triggers specific conditions, actions, or REST connector calls.

Each control has possible events you can define for it. The Cost input text field, for example, may fire an event when the form loads (On Load), an end user changes the control's value (On Change), the cursor is in the control (On Focus), the control loses focus (On Blur), or the form is submitted (On Submit).

- Select **On Change** as the event type and click **Edit**.

The event window displays, with its type listed as *On Change*. Notice how the square buttons contain +s, because you use them to either add an action, if condition, or connector call. Actions display in red, and conditions display in blue.

- Create a condition and actions for the event.
 - Click the blue outlined **+If** button to add a condition. Now solid blue round buttons display to use to configure the condition.
 - Below the solid blue **If** button, configure the If condition as follows. Note that related fields display, depending on the selections you make.

- In the **Control Name** field, select **Cost**.
 - In the **Property** field, select **Value**, then select **is greater than** below it.
 - In the **Value** field, enter 500.
- c. Next to the solid blue **Then** button, click the **+Action** button.
 - d. Below the solid red Action button that displays, configure the Then condition:
 - In the **Control Name** field, select **Justification**.
 - In the **Action** field, select **Required**.
 - e. Scroll down. Next to the solid blue **Else** button, click the **+Else Action** button.
 - f. Below the solid red Action button, configure the Else condition, which specifies what happens if the condition isn't met:
 - In the **Control Name** field, select **Justification**.
 - In the **Action** field, select **Optional**.
- The event is complete: **If** the end user enters a cost over 500, **Then** the Justification field is required, **Else** the Justification field is optional.
- g. Click **OK** to close the event window, then **Save**.

On Change

If + Condition

Type Control Name Property
Control Cost Value
is greater than

Type Value
Constant 500

Then + Action

Action

Control Name Action
Justification Required

Else + Else If + Else Action

Action

Control Name Action
Justification Optional

+ Action + If

Cancel OK

The screenshot shows the Oracle Forms configuration interface for creating a trigger. The trigger is named 'On Change'. It contains an 'If' condition where the 'Control' type is compared to 'Cost' using the 'Value' property, and the condition is 'is greater than' a constant value of 500. There are two 'Action' sections. The first action is set to 'Required' for the 'Justification' control. The second action is set to 'Optional' for the same control. At the bottom, there are 'Cancel' and 'OK' buttons.

Try the Form in Runtime

Let's try using the form as an employee and as an approver.

1. Click **Test**. A Test Application tab opens.
2. Click **Deploy**. Leave the **Add Me to All Roles** field checked and click **Deploy**. A message displays that the application deployed successfully.
3. Click **Try in Workspace**.

4. In Runtime, click your Request Travel application under **Start Application**. The Travel form displays. Notice that it's the Employees presentation (green) that you created.
 - a. Notice built-in validation for some fields. For example, if you type an invalid character in the date field, an error displays when you change to another field.
 - b. Enter a cost over 500 in the Cost field and change to another field. Notice that an asterisk displays in the Justification field to indicate that it's required.
 - c. Click **Submit**. Now an error indicates that a value is required. Enter a justification and submit again.
5. Click **Tasks** to view the task as an approver.

Notice that when you open the task, the yellow Approvers presentation displays. Notice that the name fields are read only, as you configured them.

Explore Advanced Form Options

Here are some other areas to explore in the web forms editor:

- **Try out the advanced controls**

Use the Advanced Palette to add advanced items such as panels, sections, tabs, and tables to your form. You can also configure controls that dynamically populate by calling an external service through a connector. See [Configuring Advanced Controls](#).

- **Reuse other forms defined in the application**

The Forms Palette lists previously created forms you can reuse in any form. Drag and drop a form onto the canvas and use in one of two ways:

- As a referenced form whose controls can't be edited.
- As single controls that you can edit, reposition, and remove after clicking **Detach**, which removes the link with the original form.

See [Reusing Forms](#).

- **Create a data first form**

This exercise led you through creating a form by adding controls, and data attributes were automatically created through auto binding. You can also select a business type defined for the application, and Process creates a form based on its data attributes. See [Creating a Web Form Based on a Business Type](#).

Positioning Controls on Forms

There are multiple ways to position controls in your web form.

Rows and Columns

By default, you can drop and position controls onto the web form from any palette in rows and columns. Rows and columns will be defined based on where you want to drop your control.

- Each row has a drop target on top and bottom.
- Each column has a drop target on left and right.

Each time you drop a control onto the canvas, the control will be placed in a new row. If you drop a control beside another control, it will be added to the same row as a new column. The column size is based on the number of controls in a row.

The row is based on a 12 column grid system, so for example if the row has 4 controls, each will use 3 columns. When more than 12 controls are added to a row, the remaining controls will be displayed below.

Panels, Sections and Tabs

You can group different controls inside a single control by using panels, sections and tabs.

A **Panel** control, by default, displays all the controls grouped in it vertically. However, you can change the properties of the panel and make it horizontal. You can do this by specifying the layout option as Horizontal in the **Layout** drop-down field of the General tab.

A **Section** control is similar to a panel, but is collapsible. You can use the section control to create groups of controls that users can expand and collapse. To show the section expanded when the form loads, check **Expand** in the General tab. Uncheck **Expand** to show the section collapsed. You can drag and drop any other controls inside a section control, including panels and other section controls.

A **Tab** control allows you to group controls in different tabs, to create a tabbed view in your web form. By default, when you drag a tab control onto your web form, one tab will be created. To add another tab click **Add tab**, the new tab will be added to the right of the existing one. Rearrange the tab order by dragging a tab to the right or left of another tab. Note that in small devices, such as mobiles, tabs will be shown as accordions to accommodate the content in the space. You can drag and drop other controls including panels, sections, tables and so on into an individual tab. When you select a tab, only the controls in the currently selected tab can be seen.

 **Note:**

If you delete a panel, section or tab, all the controls grouped inside the panel, section or tab get deleted.

See [Placing Controls in Panels, Sections, or Tabs](#).

Tables and Repeatable Sections

Use **Table** controls to conserve space within a web form. The table control allows you to arrange the form controls in a grid pattern. You can add columns into the table and add one control per column. When you delete a column in the table, all the controls in the column get deleted.

If auto binding is checked when you drop a table onto the canvas, a new entry in the data definition will be generated. That attribute, will be an array of type object. Every time a control is dropped to a column, an attribute inside that list will be added.

If auto binding isn't selected, you can select an attribute from the data definition. The controls inside a table can be bound to any attribute in the form data definition. When rendering the form the data will be fetched and showed for that control, but modifying the value in one row won't update every row. Also when submitting the form, the last control bound to that attribute will be the one updating the payload.

You can also select an external source to get the information from the table. When you do this, the information from the binding will be skipped, and the data will be fetched through a connector call. When submitting the form, the payload will be modified with the latest data in the table, that is, connector fetched data and user changes.

Use **Repeatable Section** controls in a web form to display multiples copies of a set of controls. This is useful when you want to allow users to enter the same type of input information multiple times in the form; for example, a phone number.

Like tables, you can configure the repeatable section controls such that information from binding can be skipped and data can be fetched through a connector call.

Unlike a table which is structured, a repeatable section is a free form container. You can add new controls on top, below or in between existing controls. You can surround an individual control with a repeatable section control or it can include an entire section control. This section control that is included in the repeatable section should contain all of the web form controls that must be repeated.

Tables and repeatable sections are a way of adding multiple information and content, dynamically to web forms. By using them, you can execute action and if triggers in the form renderer to display dynamically changing information.

See [Configuring Tables](#) and [Configuring Repeatable Sections](#).

Responsive Layout for Different Devices

The same web form with the same set of controls will be rendered in different ways for different devices. For example, if you place controls in your form for a medium device in a certain way, the same controls may be rendered in a different way for a small, large or an extra large device.

Select the device in which you want the form rendered from the options available in the **Displaying For** field. Click **Preview** to see how your form will be displayed in the selected device.

You can also specify absolute column sizes for different device sizes by deselecting **Automatic column sizes** in the Styling tab. Enter a number from 1 to 12 in the available fields to define the column sizes for different devices.

Working with Presentations

A presentation is a single view of the web form.

Presentations are useful:

- When web forms are rendered in different devices
- To present different views of the same data for specific users

When a web form is created, it will use a default presentation. When you add a new control to the web form, the new control will be added to the currently selected presentation.

Selecting, adding, or deleting a presentation in the web forms editor

1. On the Form tab, notice that the **Presentations** table lists the available presentations. When a web form is created, it uses the **Main** presentation by default.

The **Presentation** tab displays the properties of the currently selected presentation. An asterisk (*) indicates a required property. You can edit the properties of the currently selected presentation and specify new values:

Field	Description
Name	Specify a name for the presentation.
Description	Specify a suitable description for the presentation.
Border Color	Select a border color from the color palette or enter a hexadecimal value in the adjacent text field.
Border Width	Specify a border width in pixels or points. For example, 9px.
Border Style	Select a border style (Solid, Dotted, or Dashed) for the presentation.
Background Color	Specify a background color for the presentation. Select a background color from the color palette or enter a hexadecimal value in the adjacent text field.
Events	Configure events for the presentation. See Adding Dynamic Behaviors to a Form .
Global Connectors	<p>Configure a global connector call to use in one or more controls within a form. Adding a global connector call is similar to adding a connector call within events. See Executing REST Connector Calls in Events.</p> <p>While configuring a global connector, you can also determine when the connector data loads into a control using the Skip Upon Load check box.</p> <ul style="list-style-type: none"> Deselect the check box (default state) to allow connector data to be populated into a control when the form loads. Select the check box to prevent connector data being populated into a control when the form loads. When you use this option, you must explicitly execute a connector refresh on the corresponding control for the data to load. <p>A global connector call can be reused across different events. Unlike local connectors calls, you do not have to configure it in each event. Instead, you can configure it once at the presentation level and use the data from it in events across different controls.</p> <p>However, a global connector call is not automatically updated. In an event window, click +Action and select the Refresh Global Connector action available for the Presentation control to trigger a new connector call. When you use this action, all other actions present within an event are executed after the global connector update.</p>

- To add a new presentation, click the **Create Presentation** icon in the **Presentations** table. In the Create Presentation window, enter a name for the presentation and an optional description. To create this new presentation based on an existing presentation, you can select an existing presentation from the **Based On** drop-down list. Select the **Make default** option to make this presentation the default presentation. Select the **Switch to this presentation** option to switch to this new presentation immediately after creation. Click **Create**.
- To delete an existing presentation, click the **Delete Presentation** icon in the **Presentations** table. If you try to delete the default presentation, the Delete Default Presentation Confirmation window will be displayed. Select a new default presentation from the drop-down list. Click **Delete**.

4. To change the default presentation, in the **Presentations** table, select the **Make this the default presentation** icon next to the presentation you want to change to default.

Notice that you can switch to a presentation using the drop-down list of available presentations on the top of the canvas.

 **Note:**

When **specifying actions** under **Presentation**, you can use the **Set Data** action to set the value of a particular data attribute used in the form. The value is added to the payload. When the event executes, all controls using this data attribute, either directly or indirectly (through the **Computed Value** option), are updated with this value.

Binding Form Data with Controls

In web forms, data and controls are decoupled. You can define your controls and data independently and then connect the controls to the data using the binding property.

In the web forms editor, on the **Forms** tab, notice that the **Auto Binding** field is selected by default. When the auto binding property is enabled, the web forms editor creates data attributes linking them to the controls as you drag and drop controls from a controls palette onto the canvas. The data attributes are listed in the Data pane. If you disable the **Auto Binding** field, then, you must create data attributes in the Data pane and link them to every unbound control using the **General** tab's **Binding** field.

On the **General** tab, notice that the web forms editor displays the **Binding** field for every control on the canvas. In the **Binding** field, you can specify an attribute for a control by selecting an option from the autocomplete list or entering a valid binding.

 **Note:**

- The name of the binding in the **Binding** field is the same as the name of the control to which it is linked when you use auto-binding.
- When a data attribute is bound—either directly or indirectly (through computed values or events)—to multiple controls, any change to the data attribute is immediately reflected in all associated controls.
Repeatable sections or tables bound by the same data attribute are also in sync, that is, addition or deletion of rows in one control is automatically reflected in the other control.

The background colors displayed for the autocomplete options are:

- Green for valid binding names
- Red for invalid binding names
- Blue for complex binding names (their children may contain valid binding names)

If a control isn't linked to an attribute or if a control is linked to an invalid attribute, an error message is displayed in the **Binding** field and an error icon is displayed in the

canvas. When a control doesn't have a valid binding, any value entered into the control in runtime is not passed on to the form payload.

To create a new attribute, click the **Add** icon in the Data pane. In the Create Attribute window, enter a name and specify whether the new attribute is a simple type (String, Number, Boolean, Date, Date Time, and Time) or a business type. If you selected **Business** type, then, select an option from the available list of business types. Optionally, enable or disable the list of values. Click **Create**.

 **Note:**

- An attribute name can only start with a lower case letter and can only contain letters, digits, and underscores. Also, an attribute name can't start with XML.
- In the Create Attribute window, if you select **Business** in the **Type** field, then, the business types defined for your application are available in the drop-down list. Notice that the business types defined for your application are listed in the **Business Types Palette**.

To delete an attribute, click the **Delete** icon adjacent to the attribute you want to delete in the Data pane. If the attribute you're trying to delete is bound to at least one control, click **Delete** in the Delete Attribute Confirmation window. Note that the web forms editor clears the binding when you delete an attribute which is bound to a control.

To edit an attribute's name, click the **Edit** icon adjacent to the attribute you want to edit in the Data pane. Enter a new name and click **OK**.

Creating a Web Form Based on a Business Type

A business type is used to organize and store related information used in your process, such as employee data.

The web forms editor's Business Types Palette lists all business types defined for your application, including auto-generated ones created from WSDL files or ICS integrations. (Note that any inner types defined in a WSDL file are not displayed.)

Drag a business type from the Business Types Palette onto the canvas, and the web forms editor automatically creates a web form based on the business type's data attributes. You can drop additional controls onto the business type and edit their properties.

In the Data pane, you can also create business type attributes using the business types defined for your application. See [Binding Form Data with Controls](#) and [Working with Business Objects](#).

Working with Stylesheets and Styling

Using stylesheets and styling, you can customize the appearance of your web form. A stylesheet applies to the entire form. Manage and control the appearance of your web form by applying an available stylesheet to the form or importing one. Styling enables

you to change a selected control's appearance. Use styling to define display-specific properties of a currently selected control in your web form.

 **Note:**

Styling applied to an individual control will override the styles applied to the control by the stylesheet.

Changing a Form's Stylesheet

To change a form's stylesheet:

1. In the Properties pane, go to **Stylesheet** and click the Lookup stylesheet icon. The CSS browser pane opens displaying all the available stylesheets.
2. Select the stylesheet that you want to apply. The selected stylesheet will be applied to your web form.
3. Optionally, import a stylesheet. Click **Upload**. The File Open dialog box opens. Select the stylesheet you want to import from your local machine and click **Open**. The selected stylesheet will get imported and you can apply it to your form.

 **Note:**

The imported stylesheet will be available in the CSS browser and can be applied to other forms.

4. Preview your form with the selected stylesheet applied to it by selecting the **Form** tab in the Preview pane. Select the **CSS** tab to preview the styles available in the selected stylesheet. Want to know how to customize each control using a style sheet? See [Customizing Web Forms Using CSS](#).

Changing the Style of a Form's Control

To change the styling of a control:

1. Select the control for which you want to change the styling. The Properties pane changes and displays the **General** and **Styling** tabs.
2. Select the **Styling** tab. You can see a number of styling options displayed for that particular control.
3. Select the styling property you want to apply to your control and specify the styling. See [Styling Properties](#).

Styling Properties

You can change a control's appearance by defining its styling properties on the **Styling** tab in the Properties pane. Use the **Styling** tab to define display-specific properties of the currently selected control.

The following table provides an alphabetical list of the properties available on the **Styling** tab.

 **Note:**

Styling properties are control specific. Not all of the properties listed below are available for every control.

Property	Description
Automatic column size	Calculates the column size for the control based on the amount of visible controls in the row. Automatic column size is selected by default. Note: In small devices like a phone, each control is displayed in one row when automatic column size is enabled. You can specify absolute column sizes for different device sizes. To do this, deselect Automatic column size and enter a number from 1 to 12 in the four available options: Small, Medium, Large, and Extra large column sizes.
Background Color	Specifies the background color of the content area in a control.
Border (Color, Width, Style, Radius)	Determines the appearance of the border in the content area of your control by the following properties. <ul style="list-style-type: none"> • Border Color: Defines the color of the border. • Border Width: Defines the width of the border. Use standard values such as 1in, 5em or 20px. • Defines the style of the border – Solid, Dotted or Dashed. • Defines the value of rounded border corners. Use standard values such as 1in, 5em or 20px.
Color	Specifies the color of the text in the content area of a control.
Control Alignment	Specifies the alignment (Left, Right or Center) of a control in the form.
Control Class Name	Allows customization when you're using a particular stylesheet (css) in the form. Apply a css class to the control by selecting a class name from the properties defined in the form's selected stylesheet.
Font Size	Defines the font size of the text in the content area of a control. The available values are: x-small, small, normal, large, x-large.
Height	Sets an absolute height for the control. Use standard values such as 1in, 5em or 20px.
Label Color	Specifies the color of a control's label.
Label Size	Specifies the size of a control's label. The available values are: x-small, small, normal, large, x-large.
Reset Inline Styles to Default	Discards all styling selections made on the Styling tab and restore settings to their default values.
Text Alignment	Specifies the alignment (Left, Right or Center) of the content in a control. This applies to controls where the user can type in texts such as Input Text control.
Width	Sets an absolute width for the control. Use standard values such as 5in, 20px or 5%.

Customizing Web Forms Using CSS

You can use cascading style sheets (CSS) to customize the look and feel of your web form elements. Modify visual attributes, such as color, font, or margin, of an element using a custom CSS code.

You can also apply basic styling to form elements using properties available in the forms editor, see [Styling Properties](#). However, styling options available through CSS are far more extensive and specific. This section details all form elements and their corresponding CSS selectors. You can style each of these selectors by modifying several properties, such as color, font, or alignment.

This section contains the following topics:

- [Styling the Form Container](#)
- [Styling Form Controls](#)
- [Form Reference Styling](#)
- [Styling for Imported Business Types](#)
- [Managing Style Dynamically](#)
- [Some Useful Styling Tips](#)

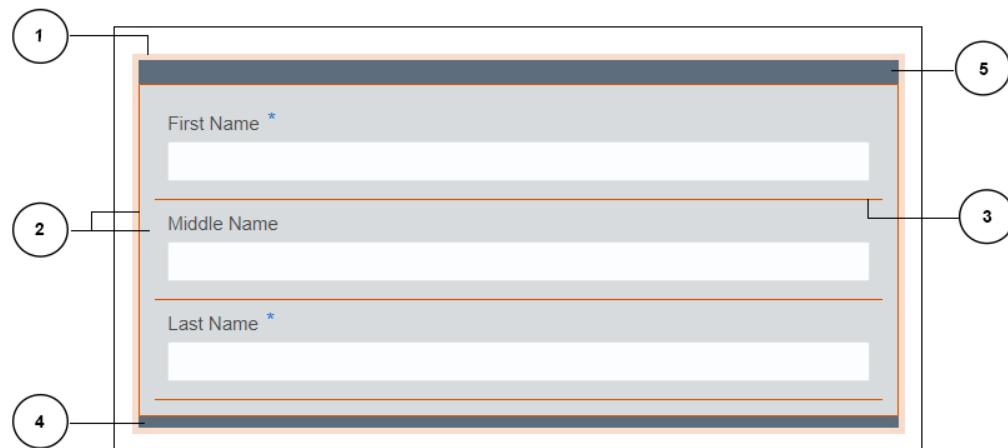
 **Note:**

This section assumes basic working knowledge of CSS.

Styling the Form Container

You can style properties that apply to the entire form, such as background or border, using selectors corresponding to the form container.

The following figure shows a sample form container with all its selectors noted:



No.	Selector Name	Description	Sample CSS Code
1	.canvas	This selector corresponds to the outermost container of a form. Use this selector to style properties like form border, padding, and so on.	.canvas { border: 5px solid #F6DDCC; }
2	.canvas__content	Use this selector to style borders and backgrounds for the content area of a form. If you have modified the styling properties of a form through the Properties pane, those styles (backgrounds or borders) are applied to this selector. Tip: To override the applied styles, use ! important in your CSS code.	.canvas__content { background: #D7DBDD !important; border: 1px solid #D35400 ! important; }
3	.row-control	Use this selector to identify a row in a form.	.row-control { border-bottom: 1px solid #D35400; }
4	.canvas__footer	Use this selector to specify footer for a form.	.canvas__footer { display: block; background: #5D6D7E; height: 10px; }
5	.canvas__header	Use this selector to specify header for a form. Note: Set the <i>display</i> property to <i>block</i> to make headers and footers visible.	.canvas__header { display: block; background: #5D6D7E; height: 20px; }

Styling Form Controls

You can style different components of a specific control using corresponding CSS selectors.

The best practice to styling a control is to define a control class name for it. See [Styling Properties](#). The control class name serves to identify the control in the CSS file, and you can use this name as a class name and modify the control's selectors within that class. See [Control Styling Example Using CSS](#).

The following table lists all customizable controls and associated selectors that you can style using CSS.

UI Control	Selectors	Description
Label (Corresponds to the title or label of all controls within a form)	.oj-label label .oj-label-required-icon .oj-label-required-icon:before .oj-label-help-icon	Specify a style for the label. Specify a style for the Required icon. You can apply properties like color, margin, padding, and so on. Specify a new signifier instead of the default Required signifier, asterisks (*). Specify a style for the Help icon.

UI Control	Selectors	Description
	.oj-label-help-icon:before	Specify a new signifier instead of the default Help icon/signifier.
Validation Message	.oj-messaging-inline-container .oj-message.oj-message-error	Specify a style for the Validation Message container.
	.oj-message-error-icon	Specify a style for the Validation Error Icon container.
	.oj-message-error-icon::before	Specify a new signifier instead of the default Error icon/signifier.
	.oj-message-content	Specify a style for the validation message content.
	.oj-message-summary	Specify a style for the validation message title.
	.oj-message-detail	Specify a style for the validation message detail.
InputText	.oj-inputtext-input	Specify a style for the Input Text container.
	.oj-inputtext.oj-invalid input	Specify a style for the Input Text container when the input is invalid.
	.oj-inputtext input[disabled]	Specify a style for the Input Text container when the control is disabled.
	.oj-inputtext input[readonly]	Specify a style for the Input Text container when the control is read-only.
TextArea	.oj-textarea textarea	Specify a style for the Text Area container.
	.oj-textarea.oj-invalid textarea	Specify a style for the Text Area container when the input is invalid.
	.oj-textarea textarea[disabled]	Specify a style for the Text Area container when the control is disabled.
	.oj-textarea textarea[readonly]	Specify a style for the Text Area container when the control is read-only.
Button	.button-control.oj-button	Specify a style for the Button control.
	.button-control.oj-button[disabled]	Specify a style for the button when it is disabled.
	.button-control.oj-button .oj-button-text	Specify a style for the button text.
	.button-control.oj-button.oj-hover	Specify a style for the button on hover (when you mouse over the button).
	.button-control.oj-button.oj-active	Specify a style for the button when it is active.
	.button-control.oj-button.oj-focus	Specify a style for the button when it is focused.
Select	.oj-select	Specify a style for the Select container.
	.oj-select .oj-select-choice	Specify a style for the selected input.
	.oj-select.oj-invalid .oj-select-choice	Specify a style for the invalid select input.
	.oj-select.oj-disabled .oj-select-choice	Specify a style for the disabled select input.

UI Control	Selectors	Description
	.oj-select-open-icon	Specify a style for the Drop-Down Icon container.
	.oj-select-open-icon:before	Specify a new signifier instead of the default Drop-Down icon/signifier.
	.oj-select-choices	Specify a style for the Select container when multiple selection is enabled.
	.oj-select-multi .oj-select-selected-choice	Specify a style for selected options when multiple selection is enabled.
	.oj-select-multi .oj-select-selected-choice-label	Specify a style for the selected option label.
	.oj-select-multi .oj-select-clear-entry:before	Specify a new signifier instead of the default Remove icon/signifier on the selected options.
Identify Browser	.oj-identity	Specify a style for the Identity container.
	.oj-identity-select	Specify a style for the identity input.
	.oj-identity.oj-invalid .oj-select-choice	Specify a style for the invalid identity input.
	oj-identity.oj-disabled .oj-select-choice	Specify a style for the identity input when the control is disabled.
	.oj-select-open-icon	Specify a style for the Drop-Down Icon container.
	.oj-select-open-icon:before	Specify a new signifier instead of the default Drop-Down icon/signifier.
	.oj-select-choices	Specify a style for the Identity container when multiple selection is enabled.
	.oj-select-multi .oj-select-selected-choice	Specify a style for selected options when multiple selection is enabled.
	.oj-select-multi .oj-select-selected-choice-label	Specify a style for the selected option label.
	.oj-select-multi .oj-select-clear-entry:before	Specify a new signifier instead of the default Remove icon/signifier on the selected options.
Checklist and Checkbox	.oj-checkboxset	Specify a style for the control container.
	.oj-choice-row	Specify a style for each check option.
	.oj-choice-row input	Specify a style for the check boxes in each row.
	.oj-choice-row label	Specify a style for the labels in each row.
	.oj-checkboxset.oj-invalid .oj-checkbox:not(.oj-disabled)	Specify a style for the invalid option.
	.oj-choice-row input[disabled]	Specify a style for the disabled option.
	.oj-choice-row-inline	Specify a style for inline options.
Radio Button	.oj-radioset	Specify a style for the control container.
	.oj-choice-row	Specify a style for each option.
	.oj-choice-row input	Specify a style for the radio button in each row.

UI Control	Selectors	Description
	.oj-choice-row label	Specify a style for the labels in each row.
	.oj-radioset.oj-invalid .oj-radio:not(.oj-disabled)	Specify a style for the invalid option.
	.oj-choice-row input[disabled]	Specify a style for the disabled option.
	.oj-choice-row-inline	Specify a style for inline options.
Number	.oj-inputnumber-wrapper	Specify a style for the Number container.
	.oj-inputnumber input	Specify a style for the number input.
	.oj-inputnumber.oj-invalid input	Specify a style for the invalid number input.
	.oj-inputnumber input[disabled]	Specify a style for the number input when the control is disabled.
	.oj-inputnumber input[readonly]	Specify a style for the number input when the control is read-only.
Date Time	.oj-inputdatetime-input-container	Specify a style for the entire Date Time control container.
	.oj-inputdatetime-input-container input	Specify a style for the input area of the Date Time control.
	.oj-inputdatetime-input-container input[disabled]	Specify a style for the Date Time control when it is disabled.
	.oj-inputdatetime-input-container input[readonly]	Specify a style for the Date Time control when it is read-only.
	.oj-inputdatetime-input-trigger	Specify a style for the Calendar and Time icons container.
	.oj-inputdatetime-calendar-icon	Specify a style for the Calendar icon.
	.oj-inputdatetime-calendar-icon:before	Specify a new signifier instead of the default Calendar icon/signifier.
	.oj-inputdatetime-time-icon	Specify a style for the Time icon.
	.oj-inputdatetime-time-icon:before	Specify a new signifier instead of the default Time icon/signifier.
Date	.oj-inputdatetime-date-only	Use this selector before the properties below to apply styling specific to the Date control.
	.oj-inputdatetime-input-container	Specify a style for the entire Date control container.
	.oj-inputdatetime-input-container input	Specify a style for the input area of the Date control.
	.oj-inputdatetime-input-container input[disabled]	Specify a style for the Date control when it is disabled.
	.oj-inputdatetime-input-container input[readonly]	Specify a style for the Date control when it is read-only.
	.oj-inputdatetime-input-trigger	Specify a style for the Calendar icon container.

UI Control	Selectors	Description
	.oj-inputdatetime-calendar-icon	Specify a style for the Calendar icon.
	.oj-inputdatetime-calendar-icon:before	Specify a new signifier instead of the default Calendar icon/signifier.
Time	.oj-inputdatetime-time-only	Use this selector before the properties below to apply styling specific to the Time control.
	.oj-inputdatetime-input-container	Specify a style for the entire Time control container.
	.oj-inputdatetime-input-container input	Specify a style for the input area of the Time control.
	.oj-inputdatetime-input-container input[disabled]	Specify a style for the Time control when it is disabled.
	.oj-inputdatetime-input-container input[readonly]	Specify a style for the Time control when it is read-only.
	.oj-inputdatetime-input-trigger	Specify a style for the Time icon container.
	.oj-inputdatetime-time-icon	Specify a style for the Time icon.
	oj-inputdatetime-time-icon:before	Specify a new signifier instead of the default Time icon/signifier.
Email	.oj-inputtext-input	Specify a style for the input container.
	.oj-inputtext.oj-invalid input	Specify a style for the input container when the input is invalid.
	.oj-inputtext input[disabled]	Specify a style for the input container when the control is disabled.
	.oj-inputtext input[readonly]	Specify a style for the input container when the control is read-only.
URL	.oj-inputtext-input	Specify a style for the input container.
	.oj-inputtext.oj-invalid input	Specify a style for the input container when the input is invalid.
	.oj-inputtext input[disabled]	Specify a style for the input container when the control is disabled.
	.oj-inputtext input[readonly]	Specify a style for the input container when the control is read-only.
Message	p, h1, h2, h3, h4, h5, h6	Use these selectors to style a message based on its type (<i>p</i> or <i>h1</i> to <i>h6</i>). Use the control class name to make it specific.
Money	.oj-inputnumber-wrapper	Specify a style for the Money container.
	.oj-inputnumber input	Specify a style for the money input.
	.oj-inputnumber.oj-invalid input	Specify a style for the invalid money input.
	.oj-inputnumber input[disabled]	Specify a style for the money input when the control is disabled.
	.oj-inputnumber input[readonly]	Specify a style for the money input when the control is read-only.

UI Control	Selectors	Description
Phone	.oj-inputtext-input	Specify a style for the input container.
	.oj-inputtext.oj-invalid input	Specify a style for the input container when the input is invalid.
	.oj-inputtext input[disabled]	Specify a style for the input container when the control is disabled.
	.oj-inputtext input[readonly]	Specify a style for the input container when the control is read-only.
Image	.image-control	Specify a style for the Image container.
Video	iframe[name=Video]	Specify a style for the Video container.
Link	.anchorLink a	Specify a style for the Link container.
Panel	No particular selector	Use the control class name in the Properties pane to build a selector.
Section and Mobile Tab	.oj-collapsible-header	Specify a style for the Section Header container.
	.oj-collapsible-wrapper	Specify a style for the section header content.
	.oj-collapsible-header-icon.oj-collapsible-open-icon	Specify a style for the Section Open icon.
	.oj-collapsible-header-icon.oj-collapsible-open-icon:before	Specify a new signifier instead of the default Open icon/signifier.
	.oj-collapsible-header-icon.oj-collapsible-close-icon	Specify a style for the Section Close icon.
	.oj-collapsible-header-icon.oj-collapsible-close-icon:before	Specify a new signifier instead of the default Close icon/signifier.
	.oj-collapsible-header p	Specify a style for the header type: paragraph.
	.oj-collapsible-header h1	Specify a style for the header type: title1 to title6.
	.oj-collapsible-header h2	
	.oj-collapsible-header h3	
	.oj-collapsible-header h4	
	.oj-collapsible-header h5	
	.oj-collapsible-header h6	
Tab	.tab-container	Specify a style for the Tabs container.
	.oj-tabs-tab	Specify a style for the tab header.
	.oj-tabs-tab.oj-selected	Specify a style for the selected tab header.
	.oj-tabs-tab.oj-disabled	Specify a style for the disabled tab header.
	.oj-tabs-title	Specify a style for the tab title.
	.oj-tabs-panel	Specify a style for the tab content.
Table	.table-container	Specify a style for the entire Table container (including the label).
	.table-control	Specify a style only for the table.

UI Control	Selectors	Description
	.oj-table-header	Specify a style for the table header section (<i>thead</i>).
	.oj-table-header-row	Specify a style for the table header row, containing all table header columns (<i>tr</i>).
	.oj-table-column-header-cell	Specify a style for the Table Column Header container (<i>th</i>).
	.oj-table-column-header-text	Specify a style for the text in each column.
	.oj-table-body	Specify a style for the table body containing all items (<i>tbody</i>).
	.oj-table-body-row	Specify a style for each row inside the table body (<i>tr</i>).
	.table-cell	Specify a style for a particular cell inside the body.
	.table-cell.oj-selected	Specify a style for the selected cell.
	button.add	Use this selector to add a button within a table.
	button.delete	Use this selector to remove a button from within a table.
Repeatable Section	.repeatable-section	Specify a style for the Repeatable Section container.
	button.add	Use this selector to add a button within a repeatable section.
	button.delete	Use this selector to remove a button from within a repeatable section.

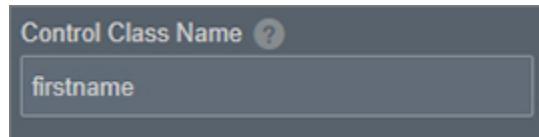
Control Styling Example Using CSS

Let's apply basic styling to a control to understand the usage of control class names and selectors. The following figure shows an Input Text control with no styling applied to it:

The image shows a simple input text field. At the top left, the label "First Name" is followed by a red asterisk indicating it is a required field. Below the label is a rectangular input box with a thin gray border. The entire control is contained within a light gray rectangular box with rounded corners.

To apply styling to this control through CSS:

1. Select the control in the form canvas.
2. In the Properties pane, select the **Styling** tab.
3. In the **Control Class Name** field, enter a name to identify the control in the CSS file as shown in the following figure:



- Now, use this name in your CSS code to modify the control's properties or selectors. The following sample code shows styling of the control's label:

```
.firstname .oj-label label {  
    font-size: 20px;  
    color: blue;  
}  
.firstname .oj-label-required-icon:before {  
    content: "#";  
}
```

- Upload this style sheet from the form's Properties pane. The customized Input Text control appears as follows:



- Similarly, you can customize other selectors associated with this control. You can also apply more than one class names to the control. In addition, you can reuse the same class name with other controls that require similar styling.

Note:

If you do not specify a control class name in the CSS code while defining styles for a selector, the styles will apply to all controls that satisfy the selector.

Form Reference Styling

When you import a form into another form by reference, the parent form's styling is automatically applied to the imported form. This behavior ensures that the specified style is enforced for a form, even when there are references.

If you want to retain the styling of the imported form, include the CSS code of the child into the parent form's CSS file. Also, make sure to use the corresponding class names in the code to apply styles specifically to the controls in the imported form.

The following example shows the CSS code of an imported form (along with a control class name) added to its parent's CSS file:

```
.oj-label label {  
    color: blue;  
}
```

```
.nominee .oj-label label {  
    color: brown;  
}
```

The resulting form styling appears as follows:

The form consists of two main sections. The first section contains fields for 'First Name' and 'Last Name', both labeled in blue. The second section, titled 'Nominee', contains two fields: 'First Name' and 'Last Name', both labeled in red. This demonstrates how a CSS class defined in the parent form's stylesheet can be applied to specific controls within a child form.

Note:

You must have added the same class name, in this case *nominee*, to the controls of the child form before using it in the parent form's CSS file.

Styling for Imported Business Types

When you drag a business type from the Business Types Palette onto the canvas, a section with a set of controls is created based on the business type's data attributes. All styles defined for the form through CSS are also applied to this auto-generated section of controls.

Managing Style Dynamically

You can manage styling of a control dynamically through the Event window. Want to know more about events? See [Adding Dynamic Behaviors to a Form](#).

Within the Event window, you can specify actions to add or remove class names to a control when a condition is fulfilled. Based on the class name assigned, the styling of the control changes during execution. To better understand dynamic styling, let us take a look at the following example.

Consider a Number control (Age) for which the allowed value is any number less than 60. Now, let us create an event to style the control based on the value entered by the user during execution.



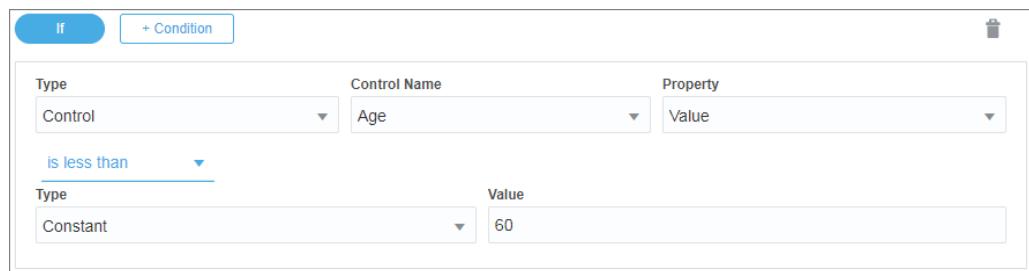
1. In the web forms editor, select the **Age** control, and locate the **Events** option in the **General** tab of the Properties pane.

2. Click the **Add** icon to define an event.

The event field changes to a drop-down menu listing event options available for the control.

3. Select the **On Change** option from the menu.

4. In the Event window, define an If condition to specify allowed values for the control as less than 60.



5. Specify actions for Then and Else parts of the condition as shown in the following figure.

The actions defined make sure that the appropriate class name is applied to the control based on the value entered by the user. Subsequently, the styling defined for the class name in CSS is dynamically applied to the control.

The screenshot shows the Oracle JDeveloper Action Editor. It contains two 'Then' sections under the 'Action' tab. Each section has three rows: 'Control Name' (Age), 'Action' (Add Class or Remove Class), and 'Control Class Name' (allowed or notallowed). The first section adds 'allowed' and removes 'notallowed'. The second section adds 'notallowed' and removes 'allowed'.

Action	Action	Control Class Name
Control Name: Age	Action: Add Class	Control Class Name: allowed
	Action: Remove Class	Control Class Name: notallowed

Action	Action	Control Class Name
Control Name: Age	Action: Add Class	Control Class Name: notallowed
	Action: Remove Class	Control Class Name: allowed

6. In this case, you can define styling for class names in the CSS file as follows:

```
.allowed .oj-inputnumber input {
color: green;
}
.notallowed .oj-inputnumber input {
color: red;
}
```

7. When the user enters a value into the control, the styling changes as follows:

- If the value entered is less than 60, the following style is rendered.

The screenshot shows an Oracle Input Number control with the label 'Age' above it. The input field contains the value '31', which is displayed in green. This indicates that the value is less than 60, and therefore the 'allowed' class is applied to the input field.

- If the value entered is greater than or equal to 60, the following style is rendered.

The screenshot shows an Oracle Input Number control with the label 'Age' above it. The input field contains the value '61', which is displayed in red. This indicates that the value is greater than or equal to 60, and therefore the 'notallowed' class is applied to the input field.

Some Useful Styling Tips

Here are some useful tips and best practices to make forms styling through CSS easy to understand, maintain, and scale.

- Follow a notation for control class names. There are different naming conventions, such as Block Element Modifier (BEM), that you can use to make class names informative and readable.
- Use the *!important* attribute effectively. This attribute allows you to override a style that has a higher specificity or priority.
- Use Scalable Vector Graphics (SVG) images for icons and backgrounds. SVG images scale correctly, avoiding any loss in quality when the form is rendered on different devices.
- Avoid using properties that alter the form layout and behavior, such as *float*, *display*, *flex*, and so on.
- Avoid fixed dimensions while styling controls. The layout of a form changes based on the device, and the content within it adjusts to the available space. If an input text control's width is set to a fixed value of 400 pixels, the control may not display properly on a phone.

Configuring Basic Controls

The Basic Palette contains a set of basic controls that you can drag onto the canvas. Note that each time you select a control on the canvas, the **General** and **Styling** tabs become available in the Properties pane to configure settings specific to the selected control. An asterisk (*) indicates a required property.

Topics

- [Configuring Text Input and Area Fields](#)
- [Configuring Buttons](#)
- [Configuring Drop-down Select Fields](#)
- [Configuring Check Lists and Check Boxes](#)
- [Configuring Radio Buttons](#)
- [Configuring Number Fields](#)
- [Configuring Date and Time Fields](#)
- [Configuring Email Fields](#)
- [Configuring Web Address URL Fields](#)
- [Configuring Message Fields](#)
- [Configuring Links](#)

Configuring Text Input and Area Fields

A text input control allows users to enter short, single line text entries. A text area control allows users to enter longer, multi-line text entries. In a text area control, as a

user enters multi-line text, if needed, a scroll bar appears to scroll down and view the entire text.

To configure a text input or a text area control:

1. From the Basic Palette, drag the **Input text** control or the **Text Area** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Placeholder	Text that will appear in the control until any text is entered. If text is removed from the control, the text specified in this field reappears.
Hint	Hint text that will display to users when a user clicks into the control.
Help	Help text that will appear for the control. If you specify help text, a help icon appears next to the name of the control and displays the help text when a user hovers over it or clicks it.
Min Length	Defines the minimum number of characters a user must enter into the control. Leave this field blank if you don't want to define the minimum length.
Max Length	Defines the maximum number of characters a user can enter into the control.
Rows	Defines the number of text rows that will be visible to a user. A scroll bar appears automatically when the number of text rows entered by the user is greater than the value specified in this field.
Pattern	Allows you to define custom validations on the type of text a user enters into the control. Enter a pattern using regular expressions. When you specify a pattern for the control, you can also specify a message in the Pattern Validation Message field that will display if the validation fails.
Events	Allows you to configure events for the control. You can assign actions or If/Then/Else conditions to the control based on a selected event. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Configuring Buttons

Use a button control to add a button to your web form.

To configure a button control:

1. From the Basic Palette, drag the **Button** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Events	Allows you to configure events for the control. You can assign actions or If/Then/Else conditions to the control based on a selected event. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Configuring Drop-down Select Fields

Use a drop-down select control to add a drop-down list to your web form.

To configure a drop-down select control:

1. From the Basic Palette, drag the **Select** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Hint	Hint text that will display to users when a user clicks into the control.
Help	Help text that will appear for the control. If you specify help text, a help icon appears next to the name of the control and displays the help text when a user hovers over it or clicks it.
Multiple	Determines whether users can select multiple values in the control. If this field is disabled, users can select only one option in the control.

Field	Description
Options Source	<p>Select a source (Static, From Data, and Connector).</p> <ul style="list-style-type: none"> • Static: Specify choices using Options Names and Options Values fields. Use Options Names to specify the label to display for an option and use Options Values to specify an internal value for an option. • From Data: In the Options List field, select a list of values options source from the data definitions available in the web form. If you selected a list of complex elements, then, in the Label Binding field, specify a data attribute that will display as the label and in the Value Binding field, specify a data attribute that will be the value. • Connector: Specify a REST connector, a resource, and an operation to use. Specify parameters to pass to the connector and define how the response should be mapped to the control properties. See Populating Controls Using REST Calls.

 **Note:**

While defining **Dynamic Behaviors** using the **Select** control, you can fetch the selected option name using the **Selected Value** option under the **Property** field.

Default Value	If you selected Static in the Options Source field, then, specify a default option in this field. If you selected From Data or Connector in the Options Source field, then, select either the first or the last value as the default value.
Events	Allows you to configure events for the control. You can assign actions or If/Then/Else conditions to the control based on a selected event. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling](#) properties.
4. Click **Preview** to try out using the control.

Configuring Check Lists and Check Boxes

Use a check list control to add a list of options to your web form. A check list allows users to select one or more options. You can add a check box control to your web form to allow users to specify a true or false value. A check box control is set to false by default. You can edit the **Default Value** property on the **General** tab to change the default value.

To configure a check list or a check box control:

1. From the Basic Palette, drag the **Checklist** or the **Checkbox** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.

Field	Description
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Help	Help text that will appear for the control. If you specify help text, a help icon appears next to the name of the control and displays the help text when a user hovers over it or clicks it.
Inline	Specify the layout of the options defined for the check list control. If enabled, the layout changes from vertical to horizontal.
Options Source	Select a source (Static, From Data, and Connector). <ul style="list-style-type: none"> • Static: Specify choices using Options Names and Options Values fields. Use Options Names to specify the label to display for an option and use Options Values to specify an internal value for an option. • From Data: In the Options List field, select a list of values options source from the data definitions available in the web form. If you selected a list of complex elements, then, in the Label Binding field, specify a data attribute that will display as the label and in the Value Binding field, specify a data attribute that will be the value. • Connector: Specify a REST connector, a resource, and an operation to use. Specify parameters to pass to the connector and define how the response should be mapped to the control properties. See Populating Controls Using REST Calls.
Default Value	If you selected Static in the Options Source field, then, specify a default option in this field. If you selected From Data or Connector in the Options Source field, then, select either the first or the last value as the default value. For the check box control, select either True or False as the default value.
Events	Allows you to configure events for the control. You can assign actions or If/Then/Else conditions to the control based on a selected event. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Configuring Radio Buttons

Add mutually exclusive radio buttons to your web form. Radio buttons allow users to select an option from a set of available options.

To configure a radio button control:

1. From the Basic Palette, drag the **Radio Button** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.

Field	Description
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Help	Help text that will appear for the control. If you specify help text, a help icon appears next to the name of the control and displays the help text when a user hovers over it or clicks it.
Inline	Specify the layout of the options defined for the control. If enabled, the layout changes from vertical to horizontal.
Options Source	Select a source (Static, From Data, and Connector). <ul style="list-style-type: none"> • Static: Specify choices using Options Names and Options Values fields. Use Options Names to specify the label to display for an option and use Options Values to specify an internal value for an option. • From Data: In the Options List field, select a list of values options source from the data definitions available in the web form. If you selected a list of complex elements, then, in the Label Binding field, specify a data attribute that will display as the label and in the Value Binding field, specify a data attribute that will be the value. • Connector: Specify a REST connector, a resource, and an operation to use. Specify parameters to pass to the connector and define how the response should be mapped to the control properties. See Populating Controls Using REST Calls.
Default Value	If you selected Static in the Options Source field, then, specify a default option in this field. If you selected From Data or Connector in the Options Source field, then, select either the first or the last value as the default value.
Events	Allows you to configure events for the control. You can assign actions or If/Then/Else conditions to the control based on a selected event. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Configuring Number Fields

Add number controls to your web form. Number controls allow users to enter decimal numbers. By default, number controls display 0 but you can set this to any decimal number using the **Default Value** field in the **General** tab.

To configure a number control:

1. From the Basic Palette, drag and drop a **Number** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Default Value	Sets a value to display to users when the form loads.

Field	Description
Placeholder	Hint text that describes the expected value. This hint text will display in the control before users enter a value.
Max	Sets a maximum value that users can enter into the control.
Min	Sets a minimum value that users can enter into the control
Step	<p>Specifies a value based on which the number will increase or decrease when users increment or decrement the value with the up or down arrow.</p> <p>For example, if the step value is set to 4, and the initial value in the number field is 0, then when users increment the value for the first time, the value will be 4. When users increment the value the second time, it will be 8 and so on.</p> <p>By default the step value is set to 1. Step must be always greater than 0 and you can't enter a value lower than 1.</p>
Events	Specifies events that trigger actions and conditions. You can customize the control's behavior by configuring events. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Configuring Date and Time Fields

Use a Date Time control in your web form. A Date Time control allow users to enter date and time together. Optionally, add a Date and a Time control to your web form to allow users to enter date and time separately. The Date Time, Date and Time controls are available in the Basic Palette.

To configure a Date Time control:

1. From the Basic Palette, drag and drop a **Date Time** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Placeholder	Hint text that describes the expected value. This hint text will display in the control before users enter a value.
Help	Help text to display to users when they hover over or click the control's help icon.
Default Value	Sets the default date and time that appears in the control when the form loads and the current control value is empty.
Max Time	Sets a maximum date and time users can enter in the control.
Min Time	Sets a minimum date and time users can enter in the control.
Format	Specifies a date format for the control. You can select from the available date formats such as yy-MM-dd, yyyy-MM-dd, MM/dd/yy, and so on.

Field	Description
Time Step	Sets the value on which the time will increase when users click the control's clock icon. For example, if the time step is set to 30 minutes and the time on the Date Time control is 12:00:00 pm, then when users click the control's clock for the first time, the time changes to 12:30:00 pm, when users click it for the second time, the time changes 01:00:00 pm, and so on.
Events	Specifies events that trigger actions and conditions. You can customize the control's behavior by configuring events. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Configuring Email Fields

Add an email control to allow users to enter a valid email address to your web form.

1. From the Basic Palette, drag and drop an email control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Default Value	Specifies an email to display to users when the form loads and the current control value is empty. You must ensure that you enter a valid email format in the field, otherwise you will get a message indicating that the email format entered isn't valid.
Placeholder	Hint text that describes the expected email format. This hint text will display in the control before users enter a value.
Hint	Useful hint text that displays to users when they select the control.
Help	Help text to display to users when they hover over or click the control's help icon.
Max Length	Sets the maximum number of characters users can enter before the @ symbol for the email.
Min Length	Sets the minimum number of characters users can enter before the @ symbol for the email.
Events	Specifies events that trigger actions and conditions. You can customize the control's behavior by configuring events. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Configuring Web Address URL Fields

Add a URL control to allow users to enter a web address URL to your web form.

1. From the Basic Palette, drag and drop a **URL** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Default Value	<p>Sets a value to display to users when the form loads and the current control value is empty.</p> <p>You must ensure that you enter a correct web address URL format in the field, otherwise you will get a message indicating that the URL format entered isn't valid.</p>
Placeholder	Hint text that describes the expected value. This hint text will display in the control before users enter a value.
Hint	Useful hint text to display to users when they select the control.
Help	Help text to display to users when they hover over or click the control's help icon.
Max Length	Specifies the maximum number of characters users can enter into the control.
Min Length	Specifies the minimum number of characters users must enter into the control.
Events	Specifies events that trigger actions and conditions. You can customize the control's behavior by configuring events. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Configuring Message Fields

Use a message control to allow users to enter a simple message to your web form.

1. From the Basic Palette, drag and drop a **Message** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Default Text	The default text message that will appear to users when the form loads.

Field	Description
Type	Sets the style and format in which the message displays, as a bold heading or as a paragraph text.
Events	Specifies events that trigger actions and conditions. You can customize the control's behavior by configuring events. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Configuring Links

Use the link control to insert a URL into a form. You can specify the value for the link URL or configure the URL value to change dynamically based on the payload. For example, a URL link that contains an order item could change based on the order ID, which could be in the payload.

1. From the Basic Palette, drag and drop a **Link** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	Defines an internal name for the control. It's an internal identifier that you'll use to identify the control.
Label	Specifies the control name that displays to a user.
Static	Select this option to use the value in the Default Label field as the control name when the form loads. This is the default selection.
Dynamic	Select this option to assign the control name dynamically when the form loads. If you select this option, you must create a data attribute in the Data pane and link it to the control using the Label Binding field. When the form loads, the value of the data attribute is fetched from the payload and assigned as the control name.

Note:

If the attribute value is not available in the payload, then the value in the **Default Label** field is used as the control name.

Label Binding	Defines a link between the control's label and a data attribute. Specify an attribute for this field by selecting an option from the autocomplete list or entering a valid binding.
Default Label	Sets a label to display to users when the form loads. The value in this field is used as the control name in the following contexts: <ul style="list-style-type: none"> • When you select the Static option in the Label field. • When you select Dynamic option in the Label field but the binding value is not available.
Value Binding	Defines a link between the control and a data attribute. Data attribute bound to the control, either automatically when Auto Binding is enabled or manually using autocompletion.

Field	Description
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Default Value	Sets a value to display to users when the form loads and the current control value is empty.
Open	Specifies whether you want the link to open in the current tab or in a new tab.
Anchor	Select this check box to enable linking to specific controls in the current form. When you select this check box, the Default Value field changes to a drop-down menu that lists names of all controls present in the form. Select a control name to which you want to link. All Basic Palette, Money, Phone, Image, and Video controls, if present in the current form, appear in the Default Value field. However, controls within repeatable sections, tables, or imported from other forms (unless detached) do not appear in the drop-down list.

 **Note:**

If the URL value for the link is set from binding or events, the value selected in the **Default Value** field is overridden.

Hide	Select this check box to hide the control. For example, you might hide a control by default, but configure another control that when selected triggers an event that displays the hidden control.
Events	Specifies events that trigger actions and conditions. You can customize the control's behavior by configuring events. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Configuring Advanced Controls

The Advanced Palette contains a set of advanced controls that you can drag onto the canvas. Note that each time you select a control on the canvas, the **General** and **Styling** tabs become available in the Properties pane to configure settings specific to the selected control. An asterisk (*) indicates a required property.

Topics

- [Configuring Currency \(Money\) Fields](#)
- [Configuring Phone Number Fields](#)
- [Including Images](#)
- [Including Videos](#)
- [Configuring Identity Browser Fields](#)
- [Placing Controls in Panels, Sections, or Tabs](#)
- [Configuring Static and Dynamic List of Values Fields](#)

- Configuring Repeatable Sections
- Configuring Tables

Configuring Currency (Money) Fields

Add a money control to your web form to allow users to enter money amounts (USD, EUR, JPY, GBP, and INR). By default, the currency type is set to **USD**. The user can change the currency type using the **Currency** property on the **General** tab. The corresponding currency symbol displays next to the currency amount. Users can use commas or decimal point to enter different amounts into the control, such as £ 30,700.00. If users don't specify commas or decimal point, the web form automatically displays these symbols. The web form rounds all currency amounts to two decimal places.

To configure a money control:

1. From the Advanced Palette, drag the **Money** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Currency	Allows you to change the currency type.
Placeholder	Text that will appear in the control until any text is entered. If text is removed from the control, the text specified in this field reappears.
Hint	Hint text that will display to users when a user clicks into the control.
Help	Help text that will appear for the control. If you specify help text, a help icon appears next to the name of the control and displays the help text when a user hovers over it or clicks it.
Min	Specify the minimum amount that users need to enter into the control.
Max	Specify the maximum amount that users can enter into the control.
Step	Specify a step value based on which the amount will be incremented or decremented correspondingly when a user increments or decrements the amount in the control. By default, the step value in this field is set to 1. For example, if the step value specified is 3 and the initial amount is \$ 0.00, then, when a user increments the amount in the control for the first time, the amount is updated to \$ 3.00. When the user increments the amount again for the second time, the amount is updated to \$ 6.00 and so on.
Show increment/decrement buttons	Select this check box to display the up and down arrow buttons used to increment or decrement the control value.
Events	Allows you to configure events for the control. You can assign actions or If/Then/Else conditions to the control based on a selected event. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).

4. Click **Preview** to try out using the control.

Configuring Phone Number Fields

Add a phone control to your web form to allow users to enter phone numbers in International or US formats. The phone control uses the US format by default and displays the expected phone number pattern (xxx-xxx-xxxx). You can change the format using the **Format** property on the **General** tab.

To configure a phone control:

1. From the Advanced Palette, drag the **Phone** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Format	Specify a phone number format. By default, the US format is selected.
Placeholder	Text that will appear in the control until any text is entered. If text is removed from the control, the text specified in this field reappears.
Hint	Hint text that will display to users when a user clicks into the control.
Help	Help text that will appear for the control. If you specify help text, a help icon appears next to the name of the control and displays the help text when a user hovers over it or clicks it.
Max Length	Defines the maximum number of characters a user can enter into the control.
Min Length	Defines the minimum number of characters a user must enter into the control. Leave this field blank if you don't want to define the minimum length.
Events	Allows you to configure events for the control. You can assign actions or If/Then/Else conditions to the control based on a selected event. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Including Images

Use an image control to include an image to your web form. You can specify an absolute or a relative image URL using the **Image URL** property on the **General** tab.

To configure an image control:

1. From the Advanced Palette, drag the **Image** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Image URL	Specify an image URL (absolute URL or relative URL).
Alternative text	Enter text to display if the image can't be loaded.
Events	Allows you to configure events for the control. You can assign actions or If/Then/Else conditions to the control based on a selected event. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Including Videos

Use a video control to add a video, such as a Youtube or Vimeo video, to your web form. You can specify full video URLs, embedded URLs, or shortened URLs using the **Source Url** property on the **General** tab. You can optionally loop the video or specify to automatically start playing the video when loaded.

To configure a video control:

1. From the Advanced Palette, drag the **Video** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Computed Value	Set this property to apply a computation to the control. See Creating Computed Controls .
Source Url	Specify a valid source URL for the video.
Allow Fullscreen	Set this property to allow users to play the video in full screen mode. By default, this field is enabled.
Loop	Set this property to loop the video continuously.
Auto Play	Set this property to automatically start playing the video when the web form loads.
Show Controls	Specify whether to display play or pause controls for the video. By default, this field is enabled.
Events	Allows you to configure events for the control. You can assign actions or If/Then/Else conditions to the control based on a selected event. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Configuring Identity Browser Fields

Add an Identity Browser control to allow users to search and select individuals to be notified or assigned tasks downstream in the process.

Use identities to search for users, groups, or roles. You can add single or multiple entries to the Identity Browser based on the configuration you select. Use options in the search filter to restrict the scope of the search.

To configure an Identity Browser control:

1. From the Advanced Palette, drag the **Identity Browser** control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Label	Specifies the control name that displays to a user.
Binding	Defines a link between the control and a data attribute.
Placeholder	Text that will appear in the control until any text is entered. If text is removed from the control, the text specified in this field reappears.
Hint	Hint text that will display to users when a user clicks into the control.
Help	Help text that will appear for the control. If you specify help text, a help icon appears next to the name of the control and displays the help text when a user hovers over it or clicks it.
Default Scope	Specifies the default scope for the identity search. Available options in the drop-down menu are All , User , Group , and Role . The value of this field is set to User by default.
Required	Set this property to require users to complete the control in order to successfully submit the form.
Multiple	Set this property to allow multiple entries to the control.
Disabled	Set this property to display the control as inactive.
Hide	Set this property to hide the control. For example, you might hide a control by default, but configure another control that when selected triggers an event that displays the hidden control.
Scope Filter	Set this property to allow filtering of results in the control.
Auto Focus	Set this property to automatically select the control when the web form loads.

Field	Description
Events	Allows you to configure events for the control. You can assign actions or If/Then/Else conditions to the control based on a selected event. See Adding Dynamic Behaviors to a Form .

 **Note:**

Except for assigning a constant value, you cannot trigger any changes to the Identity Browser using other controls. Actions such as assigning function results, connector call values, or data from the payload or another control are not applicable in the context of Identity Browser.

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Click **Preview** to try out using the control.

Retrieving a List of User Identities

An identity object contains several fields, such as title, ID (user identity), type, e-mail, first name, last name, and contact number.

Using a combination of Identity Browser and Text controls, you can retrieve a comma-separated list of user identities (IDs), which can be used by other process elements.

To fetch a list of IDs:

1. Add an **Identity Browser** control to the form canvas.
2. Edit the control's properties as required. Make sure to select the **Multiple** checkbox to allow multiple entries to the control.
3. Drag the **Input text** control onto the canvas. You may also use the **Text Area** control.
4. Configure an event to populate data in the **Input text** control field based on the data entered into the **Identity Browser** control. See [Adding Dynamic Behaviors to a Form](#) for details on creating events.
 - a. Select **Identity Browser**, and in the Properties pane's **General** tab, scroll down until you see **Events**.
 - b. Click **Add**  to define an event.
 - c. Select an event option, for example, **On Change**; this event is activated when an end user changes the control's value.
 - d. After selecting the event option, click the editing icon next to the events dropdown menu to specify the action. An event window appears.
 - e. In the event window, click **+Action** to add an action.
 - A solid red **Action** indicator appears. Below it, configure the action.
 - f. In the **Control Name** field, select the control the action will affect, in this case select **InputText**.
 - g. In the **Action** field, select **Value**.



Placing Controls in Panels, Sections, or Tabs

You can add panels, sections and tabs to your web form and use them to group multiple controls under a single control.

To configure a panel, section or tab control:

- From the Advanced Palette, drag and drop a **Panel**, **Section** or **Tab** control onto the canvas.
 - A text *Drop Elements to this panel!* indicates that you can drop controls into the panel.
 - A text *Drop Elements to this section!* indicates that you can drop controls into the section.
 - A text *Drop Elements to this tab!* indicates that you can drop controls into the tab. By default the tab control displays one tab (Tab1) and an **Add Tab** option besides it. Add more tabs to your tab control by clicking **Add Tab**.
 - Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Layout	For a panel control, specifies whether the controls inside it should be displayed vertically or horizontally. This is a mandatory field for the panel control.
Type	For a section control, sets the style and format in which the section label displays.
Expand	For a section control, specifies if the section control is expanded or collapsed when the form loads. By default, this field is checked.
Event	Specifies events that trigger actions and conditions. You can customize the control's behavior by configuring events. See Adding Dynamic Behaviors to a Form .

3. On the Styling tab, edit the control's [styling properties](#).
4. Drag and drop individual controls from the basic or Advanced Palette into the panel, section or tab control.
See [Positioning Controls on Forms](#).
5. Configure general and styling properties for the controls inside your panel, section or tab control.
See [Configuring Basic Controls](#) and [Configuring Advanced Controls](#).

 **Note:**

If you delete the panel, section or tab control, all the controls grouped under it get deleted.

6. Click **Preview** to try out using the control.

 **Note:**

If the data in a field within a section or tab is invalid, a validation error is displayed for the field and the entire section or tab is marked invalid.

Configuring Static and Dynamic List of Values Fields

Use checklists, radio button or select controls to create static or dynamic list of values in your web form. Static list of values are defined in the control properties. Dynamic list of values are defined from a collection of available attributes or are based on a REST call.

You can configure static and dynamic list of values for checklists, radio buttons and select controls from **Option Source** in the **General** tab. In **Option Source** specify whether the control's options should come from static values you enter, from a list of values data attribute, or externally from a REST connector created for the application.

To configure static and dynamic list of values (for checklist, radio button and select controls):

1. From the Basic Palette, drag and drop a checklist, radio button or select control onto the canvas.
2. Select the control and configure its static list of values on the Properties pane **General** tab.
 - a. Go to **Option Source** and click **Static**.
 - b. In the **Option Names** field, enter a label to display for each option.
 - c. In the **Option Values** field, enter an internal value for each option.
 - d. Specify a default value from the option values available in the **Default Value** field.
 - e. Select an option in the **Autofocus** field to make that option the selected option when the form loads.
3. Optionally, configure dynamic list of values for the checklist, radio button or select control. There are two options to configure dynamic list of values.
 - Specify that the control's options should come from a list of value data attributes defined in the Data definition pane.
 - a. Click **From Data**.
 - b. Select the list of values options source from available attributes in the **Options List** field.
 - c. Specify a default value from the options available in the **Default Value** field.
 - d. Select an option in the **Autofocus** field to make that option the selected option when the form loads.
 - Specify that the control's options should come from a REST connector created for the application.
 - a. Click **Connector**.
 - b. Specify the connector settings in the Connector, Resource and Operation fields and map response settings. See [Populating Controls Using REST Calls](#).
 - c. Select an option in the **Autofocus** field to make that option the selected option when the form loads.
4. Click **Preview** to try out using the control.

Configuring Repeatable Sections

Use repeatable section controls to display multiple copies of a set of controls in your web form. You can use repeatable section controls to create dynamic content for your web form.

To configure a repeatable section control:

1. From the Advanced Palette, drag and drop a repeatable section control onto the canvas.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Binding	Defines a link between the control and a data attribute.
Label Binding	Allows you to define a dynamic label. To do this, you can use a data attribute listed in the Data definition pane.
Users can Add/ Remove Rows	Allows users to add or remove rows to the repeatable section.
Multiple Selection	<p>Allows users to select multiple rows of the repeatable section.</p> <p>While using a repeatable section with multiple-selection enabled to define Dynamic Behaviors or Computed Values, the options available (specific to multiple selection) in the Event or Computed Value window under the Which? field are listed here:</p> <ul style="list-style-type: none"> • For Each Selected: Use to retrieve a value from another control to apply to each selected row of a repeatable section. • All Selected: Use to retrieve values from all selected rows of a repeatable section to apply to another control. <p>Similarly, in a repeatable section with two controls in a row, you can use these options to apply values from one control to another on occurrence of an event. In this case, when you chose to apply an action to each selected row (by selecting the For Each Selected option), an additional option is available to chose the value source:</p> <ul style="list-style-type: none"> • Current Iteration Row: Use to retrieve the value from one control of the current selected row and apply to another control in the same row.
Use Data from Connector	<p>Allows you to populate the repeatable section from a REST connector defined for the application.</p> <p>Specify the connector settings in the Connector, Resource and Operation fields and map response settings. See Populating Controls Using REST Calls.</p>
Events	Specifies events that trigger actions and conditions. You can customize the repeatable section control's behavior by configuring events. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#).
4. Drag and drop individual controls from the Basic or Advanced Palette into the repeatable section control.

See [Positioning Controls on Forms](#).

5. Configure general and styling properties for the controls inside your repeatable section control.

See [Configuring Basic Controls](#) and [Configuring Advanced Controls](#).

 **Note:**

If you delete the repeatable section control, all the controls grouped under it get deleted.

6. Click **Preview** to try out using the control.

Configuring Tables

Use table controls to group multiple controls in a grid pattern into your web form. You can use table controls to create dynamic content for your web form.

To configure a table control:

1. From the Advanced Palette, drag and drop a table control onto the canvas.
By default the table control contains one column.
2. Select the control and edit its properties on the Properties pane's **General** tab.

Field	Description
Name	An internal identifier that you will use to identify the control.
Binding	Defines a link between the control and a data attribute.
Columns	In the Columns field, click Add  to add columns. Edit the label of the columns in the column fields. Click the delete icon beside the column field to delete a column.
Users can Add/ Remove Rows	Allows users to add or remove rows to the table.
Multiple Selection	Allows users to select multiple rows of the table. While using a table with multiple-selection enabled to define Dynamic Behaviors or Computed Values , the options available (specific to multiple selection) in the Event or Computed Value window under the Which? field are listed here: <ul style="list-style-type: none"> • For Each Selected: Use to retrieve a value from another control to apply to each selected row of a table. • All Selected: Use to retrieve values from all selected rows of a table to apply to another control. Similarly, in a table with two columns, you can use these options to apply values from one column to another on occurrence of an event. In this case, when you chose to apply an action to each selected row (by selecting the For Each Selected option), an additional option is available to chose the value source: <ul style="list-style-type: none"> • Current Iteration Row: Use to retrieve the value from one cell of the current selected row and apply to another cell in the same row.
Use Data from Connector	Allows you to populate the table from a REST connector defined for the application. Specify the connector settings in the Connector, Resource and Operation fields and map response settings. See Populating Controls Using REST Calls .
Events	Specifies events that trigger actions and conditions. You can customize the control's behavior by configuring events. See Adding Dynamic Behaviors to a Form .

3. On the **Styling** tab, edit the control's [styling properties](#). You can specify column width in several ways:
 - Leave the **Automatic column size** checkbox selected so the table's columns automatically resize to fit the device on which the form is viewed.

- Deselect the **Automatic column size** checkbox and specify column sizes for different device sizes by entering a number from 1 to 12 in the column size fields that display for different devices.
 - In the **Table Columns Width** fields, specify an absolute width for each column such as 2in (inches), 5cm (centimeters), 100px (pixels), or 25% (percent). If the column widths together exceed the table width, a scroll bar displays, unless columns are defined with percentages. If percentages are specified but exceed 100%, column widths are displayed proportionally across the table.
4. Drag and drop individual controls from the Basic or Advanced Palette into the columns.
- Each column can have one control. See [Positioning Controls on Forms](#).
5. Configure general and styling properties for the controls inside your table control. See [Configuring Basic Controls](#) and [Configuring Advanced Controls](#).

 **Note:**

If you delete the table control, all the controls grouped under it get deleted.

6. Click **Preview** to try out using the control.

Creating Computed Controls

Within a web form, you can define a control as a computed field that uses a data definition, value of an existing control, result of a custom formula, or connector data as its value.

You can define complex formulas using different functions based on data definitions, connector data, and existing controls to compute the control value. All controls that support binding also support computed values.

To convert an editable control to a computed control in a web form:

1. Click on the control in the form's canvas and select the **Computed Value** check box on the Properties pane's **General** tab. An **Edit** button appears.
2. Click **Edit** to define how the control's value is calculated.
3. In the Configure Computed Value window, select from the following options in the **Type** field:
 - a. **Constant** lets you specify a value for the selected control. Be sure to specify the corresponding value in the **Value** field.
 - b. **Data Definition** lets you use the data from the payload. When you click the adjacent **Value** field that displays, available items are listed.
 - c. **Control** lets you use the data from a control. Be sure to select the control name.
 - d. **Function** lets you define custom formulas using common operations with strings, values and arrays, as described in [Specifying Functions](#).
 - e. **Connector Data** lets you assign a value from a global connector call.

4. After configuring the value, click **OK**, then **Save**.

 **Note:**

- If a control has a computed value and also has a value assigned through payload, the computed value has the priority.
- If a control has a computed value but is edited, either through user input or event actions, the new value remains. However, when a dependency (data definition, control value, or connector data) is updated, the new computed value is set.

Example of a Computed Control

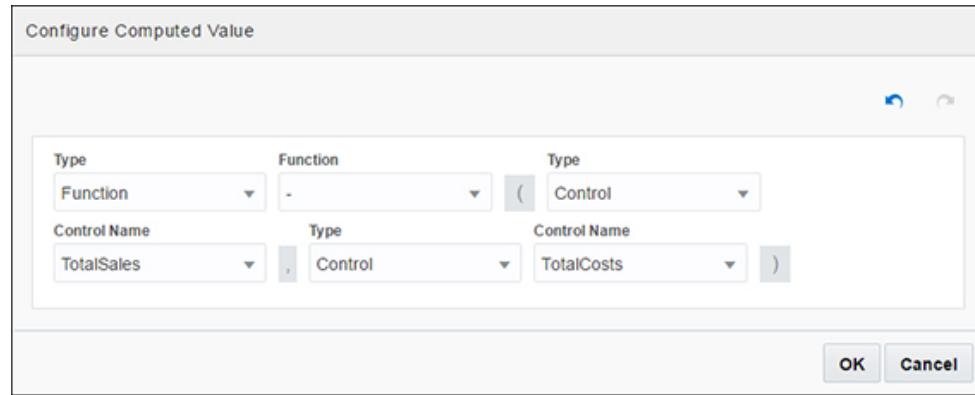
Let's create a web form to collect sales and costs data through user input and calculate total profits. To create a web form from scratch, see [Creating Web Forms](#).

When the form loads, you should be able to enter sales and costs data in editable controls, and a computed control should calculate the difference between data entered to arrive at total profits.

Now, let's add and configure controls in the web form to suit this example.

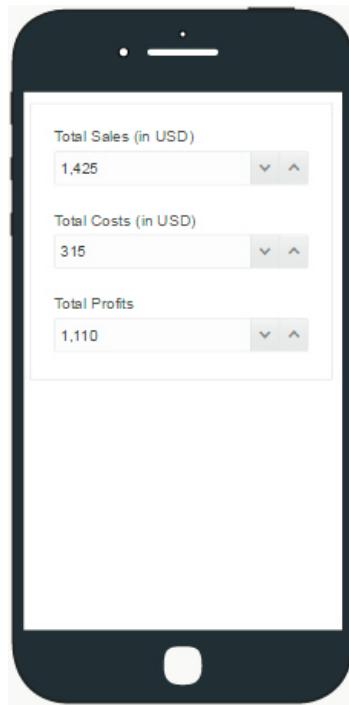
1. Add and configure two editable number fields onto the form's canvas.
 - a. From the Basic Palette, drag and drop a **Number** control onto the canvas.
 - b. Drag and drop another **Number** control below the first control.
 - c. Select the first **Number** control. On the **General** tab, change the **Name** field to *TotalSales*, and the **Label** field to *Total Sales (in USD)*.
 - d. Repeat the previous step to change the second control's name to *TotalCosts* and its label to *Total Costs (in USD)*.
2. Add a third number field and configure it as a computed control.
 - a. Drag and drop a **Number** control below the previously added controls.
 - b. Select the control. On the **General** tab, change the **Name** field to *TotalProfits*, and the **Label** field to *Total Profits*. Select the **Computed Value** check box and click the **Edit** button.
 - c. In the Configure Computed Value window, define the formula to calculate profits.
Select **Function** in the **Type** field, and choose the **Subtract (-)** function in the **Function** field.

As illustrated in the following figure, specify parameters for additional fields to use values from editable controls.



Click **OK** to close the window.

3. Click **Preview** to test the computed control. The following figure shows sample data entered and profit calculated automatically.



Localizing Web Forms

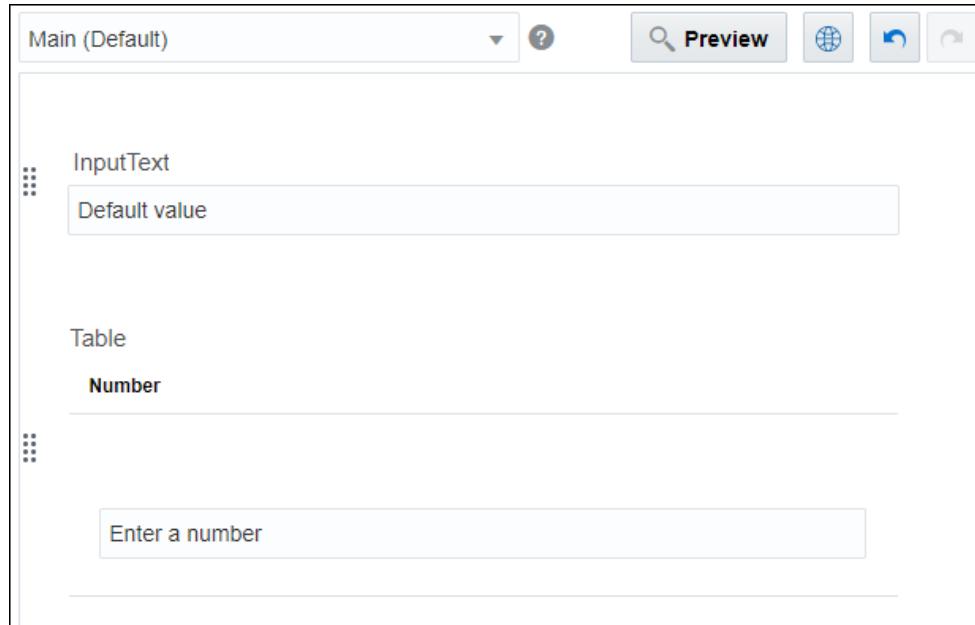
If you have added a language to localize an application's content, you can supply localized strings for control labels and control fields that take constants for input (default value, hint text, and so on) in the forms editor.

When the application is published, users can view the localized version of the form in runtime by choosing their preferred language.

To add a new language to an application, see [Localizing Applications](#). When you have added a language in addition to the default (English) language, the **Translate** (🌐) button appears in the forms editor. Click this button to supply localized strings for controls present within the form.

Let's consider a simple example to understand this feature in detail.

1. Create a web form with a couple of controls as shown in the following figure:



2. Click **Translate** (🌐) to provide localized strings corresponding to these controls.
3. In the Translate form window, select a language for which you want supply localized strings from the **Languages** drop-down field.
All languages added to the application are present in the **Languages** drop-down field.
4. Specify the strings for each control in the form and click **OK**. The following image shows strings provided for the Spanish language:

Note:

In addition to control fields, you can also provide localized strings for constant values that you specify through events or as computed values.

Translate form

ID	Default Value	Current Language
InputText - Label	InputText	Texto
InputText - Default Value	Default value	Valor por defecto
Table - Label	Table	Tabla
Column - Label	Number	Número
InputText1 - Label	InputText1	Texto1
InputText1 - Default Value	Enter a number	Ingrese un número

OK **Cancel**

5. If a language has a sub-locale corresponding to a country, for example, Spanish (Argentina), you can specify strings that differ for this sub-locale. The remaining strings are picked up from the parent language. The following figure shows example strings specified for a sub-locale:

Translate form

ID	Default Value	Current Language
InputText - Label	InputText	Ingrese Texto
InputText - Default Value	Default value	
Table - Label	Table	
Column - Label	Number	
InputText1 - Label	InputText1	Ingrese Texto1
InputText1 - Default Value	Enter a number	

 Note:

- If you have added a new language to the application but not specified localized strings for it in the forms editor, the default strings (in English) for each control are displayed when a user selects this language in runtime.
- If you have added a new language with a sub-locale to the application but not specified localized strings for it in the forms editor, the parent language strings (if specified) or default strings for each control are displayed when a user selects this sub-locale in runtime.

6. In the forms editor, you can also specify strings for languages that have unique scripts, such as Chinese, Arabic, or Hindi.

ID	Default Value	Current Language
InputText - Label	InputText	ଇନ୍‌ପୁଟ ଟେକସ୍ଟ

7. If you add a new control to the forms editor, the Translate form window is automatically updated to show this control's label. However, other fields that take constants for input are displayed only if you have entered a value in these fields.
8. If you delete a control from the forms editor, the localized strings that were specified for the control show up as grayed out rows in the Translate form window. To completely remove a string of a deleted control, click the corresponding **Delete** button or click **Clean** to clean up the entire window.

Translate form		
ID	Default Value	Current Language
Table - Label	Table	Tabla
Column - Label	Number	Número
InputText1 - Label	InputText1	Texto1
InputText1 - Default Value	Enter a number	Ingrese un número
Unknown	Unknown	Texto
Unknown	Unknown	Valor por defecto

9. Click **Preview** to see how the form is displayed for different language selections.



10. Click **Save** to save your changes to the form.

Previewing Forms and Their Payload

At any point in configuring a web form, preview it to see its appearance at different device sizes, test its behavior, and view its payload.

1. In the web form editor, click **Preview**.
2. In the Preview window, switch to device sizes users might use to see your web form's layout. The web forms automatically adjust their layout according to the screen size of the device that users use to display them. Use the drop-down fields to change the language and display percent (for example, choose 150% to zoom in on a form).
3. Enter sample values in the form's controls, and click **Submit**.
Notice that the field at the bottom of the Preview window displays the payload values. The payload values are the values that you specified in the web form's controls. These values are stored as data attributes for use in a process.
Dynamically populated controls, such as controls that call a REST connector, also function in preview mode.
4. Click **Close**.

Adding Dynamic Behaviors to a Form

Use events to introduce dynamic behaviors into your web forms, and combine them with actions, conditions, functions, and REST connector calls.

For example, you can introduce the following behaviors into your forms:

- Populate data in a control field based on another control field in the form. For example, a Country select field will impact the State select field and the State select field will impact the City select field.

- Enable control field validation based on another form control field. For example, if Start Date is given then End Date is mandatory or a Full Name gets its value from the First Name and Last Name.
- Make a REST call on demand, store the call's response, and use response data in an event action or condition.

 **Note:**

To use logged-in user's credentials when loading a form to execute a REST operation, define a REST connector without credentials. To use the same operation as a service call in the process, define another identical REST connector with credentials. This applies to internal REST API calls only.

Topics

- [Configuring Events](#)
- [Specifying Actions](#)
- [Specifying Conditions](#)
- [Specifying Functions](#)
- [Executing REST Connector Calls in Events](#)
- [Populating Controls Using REST Calls](#)
- [Linking and Refreshing List of Value Fields](#)

Configuring Events

By configuring one or more events on a control or presentation, you change the control or presentation's behavior. Configuring events in forms enables you to trigger connector calls, actions, conditions, and functions.

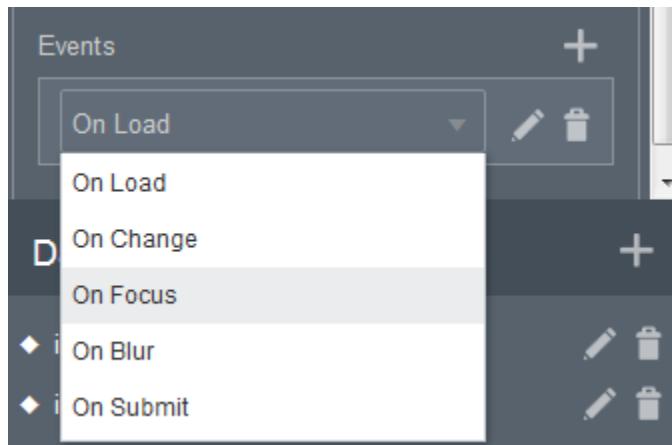
To configure events for a control or presentation:

1. In the web forms editor, select a control or presentation, and locate the **Events** options.
 - a. For a control, select the control and scroll down the **General** tab of the Properties pane until you see **Events**.
 - b. For a presentation, click a blank area of the form so that the Form and Presentation tabs display in the Properties pane. On the Form tab, select a presentation, then select the **Presentation** tab. Scroll down until you see **Events**.

Notice placeholder text displayed in the Events field *No events defined*.

2. Click **Add**  to define an event.

The event field changes to a drop-down menu listing event options available for the control or presentation.



3. Select an event option.

Available event options are specific to the selected control or presentation.

Event	Description	Control or Presentation
On Load	Fires when a form or presentation loads.	Input Text, Text Area, Date, Time, Date Time, Email, URL, Money, Phone, Checklist, Checkbox, Radio Button, Button, Select, Section, Table, Repeatable Section, Message, Image, Video, Tab and Panel. Presentation
On Change	Fires when an end user changes the control's value.	Input Text, Text Area, Date, Time, Date Time, Email, URL, Money, Phone, Checklist, Checkbox, Radio Button and Select.
On Focus	Fires when the cursor is in the control.	Input text, Text Area, Number, Date, Time, Date Time, Email, URL, Money and Phone.
On Blur	Fires when the control loses focus.	Input text, Text Area, Number, Date, Time, Date Time, Email, URL, Money and Phone.
On Submit	Fires before the form or presentation is submitted.	Input Text, Text Area, Date, Time, Date Time, Email, URL, Money, Phone, Checklist, Checkbox, Radio Button, Button and Select. Presentation
On Click	Fires when an end user clicks the button.	Button and Link
On Expand	Fires when a section is expanded.	Section
On Collapse	Fires when a section is collapsed.	Section
On Expand Toggle	Fires when an expanded section is toggled.	Section
On Row Add	Fires when an end user adds a row to a table or repeatable section.	Table and Repeatable Section
On Row Remove	Fires when an end user removes a row from a table or repeatable section.	Table and Repeatable Section

Event	Description	Control or Presentation
On Selection Change	Fires when an end user changes the row selection in a table or repeatable section.	Table and Repeatable Section

4. After selecting an event option, click the editing icon next to the events drop-down menu to specify the actions, conditions, connectors, or functions for the event. Click **OK** to complete configuring the event.

See [Specifying Actions](#), [Specifying Conditions](#), and [Executing REST Connector Calls in Events](#).

 **Note:**

To cancel an event, click its delete icon. Click **Add**  to define an additional event. You can define multiple events for a control.

Specifying Actions

Actions let you trigger changes to a control. You can choose from a variety of control and style actions. For example, you might configure a Clear button to clear the values of other form controls.

To specify actions for a control:

1. Select a control on the form canvas and specify an event for it. For example, choose **On Click** for a button control. See [Configuring Events](#).
2. Click the **Event** editing icon adjacent to the **Event** field.

The event window displays with the selected event option (for example, On Load) at the top, and color coded buttons for adding actions, conditions, or connectors to the event.

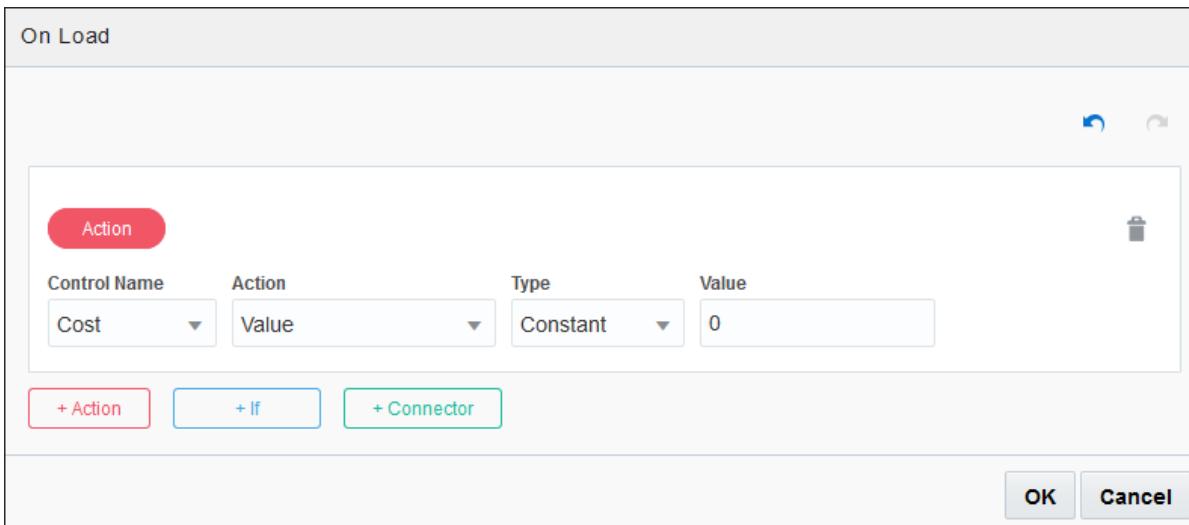
3. Click **+Action** to add an action.

A solid red **Action** indicator appears. Below it, configure the action.

4. In the **Control Name** field, select the control the action will affect.
5. In the **Action** field, select the action to take place. For example, select **Clear Value** to clear the selected control's value.

You can select from a variety of actions grouped in **Control** and **Style** categories. The options vary based on the control.

6. If configuring a control action, complete any additional fields that display. Control actions such as Value, Label, Help, Hint, Placeholder, Min Value, Max Value, and Pattern require further configuration for **Type** and **Value** fields.



If a **Type** field displays, select from these options:

- **Constant** lets you specify a value for the selected control. Be sure to specify the corresponding value in the **Value** field.
- **Data Definition** lets you use the data from the payload. When you click the adjacent **Value** field that displays, available items are listed.
- **Control** lets you use the data from a control. Be sure to select the control name and property to use.

Properties are control specific. For example, some controls, such as the number control, have many properties as options. Other controls, such as the image control, only have one property: Hidden. Email and phone controls have a property Placeholder but don't have the Min Value and Max Value properties.

- **Function** lets you perform common operations with strings, values and arrays, as described in [Specifying Functions](#).
- **Connector Data** lets you assign a value from a connector call made from the same event.

 **Note:**

You can modify multiple properties of controls such as their value or label. You can also use the data, such as connector data, to modify the control, but you cannot modify data.

7. If configuring a style action, complete additional fields that display for the selected action.

For example, you might combine a condition with a style action so that if a certain value is exceeded, a control's color, label, or class changes to alert users.

8. If needed, use the buttons at the bottom of the event window to configure additional actions or specify a condition or connector call. See [Specifying Conditions](#) and [Executing REST Connector Calls in Events](#).

Connector calls cannot be executed based on a condition. Connectors are always executed when their event is executed. You can later use the connector response values in any condition or action.

 **Note:**

In the event window, use the **Undo** or **Redo** buttons to remove or restore recent changes to actions. Use the delete icon to delete an action. Click **Cancel** to exit the event window without saving changes.

9. After completing the event, click **OK**, then **Save**.

Specifying Conditions

Conditions let you configure an If/Then/Else condition to trigger an action or connector call for a control's selected event or a control's specified action.

To specify conditions for a control:

1. Select a control on the form canvas and specify an event for it. See [Configuring Events](#).
2. Click the event editing icon adjacent to the **Event** field.

The event window displays with the selected event option (for example, On Load) at the top, and color coded buttons for adding actions, conditions, or connectors to the event.

3. Click **+If** to add a condition.

Multiple items display to help you construct and complete the condition.

- Solid blue **If**, **Then**, and **Else** indicators signal each portion of the condition to complete.
- Use the **+Condition**, **+Action**, **+Else If**, and **+Else Action** buttons to add additional conditions or actions to the condition.
- Use the fields displayed under the **If** or **Else If** indicators to define conditional behavior.

4. Complete the If portion of the condition.

- a. Select a type from the **Type** field and complete fields that display. Select from these options:

- **Constant** lets you specify a value for the selected control. Be sure to specify the corresponding value in the **Value** field.
- **Data Definition** lets you use the data from the payload. When you click the adjacent **Value** field that displays, available items are listed.
- **Control** lets you use the data from a control. Be sure to select the control name and property to use.

Properties are control specific. For example, some controls, such as the number control, have many properties as options. Other controls, such as the image control, only have one property: Hidden. Email and phone controls have a property Placeholder but don't have the Min Value and Max Value properties.

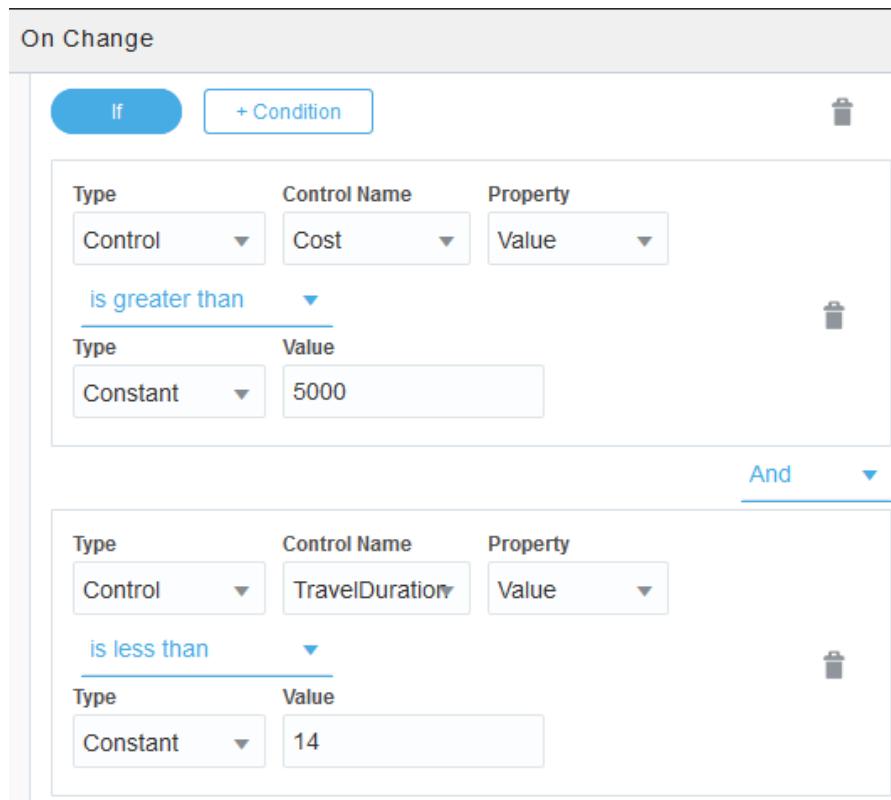
- **Function** lets you perform common operations with strings, values and arrays, as described in [Specifying Functions](#).
 - **Connector Data** lets you assign a value from a connector call made from the same event.
- b. Select an operation. For example, choose **is True** from the drop-down field.

 **Tip:**

To check for a null value, choose **is False**; if a control has no value, the condition result returns true. Conversely, choose **is True** to check that a property is present.

- c. Based on the selected operation, complete a second **Type** field and additional fields that display. You must specify a type and additional fields for all operations except when **is True** or **is False** are selected.
- d. To add another condition, click **+Condition**, specify if it's an **And** or an **Or** condition and configure it.

For example, as shown in the figure, you can configure a condition for the Cost control field of a travel request form such that if cost is greater than 5000 and the duration of travel is less than 14 days then you can set further Then or Else conditions to be triggered.



The screenshot shows the 'On Change' configuration interface. It displays two conditions stacked vertically under an 'And' logical operator. Each condition is defined by three dropdown menus: Type (Control), Control Name (Cost/TravelDuration), and Property (Value). Below each property dropdown is a comparison operator ('is greater than'/'is less than') and a value input field ('5000/14'). The interface includes a '+' Condition button to add more conditions and a trash can icon to delete existing ones.

5. Complete the Then portion of the condition by clicking **+Action** and completing the action. See [Specifying Actions](#).

For example, for the Cost control of a travel request form if the cost is greater than 5000 and the number of days of travel is less than 14 then the Justification field is required in the form.

The screenshot shows the Oracle Forms configuration interface for dynamic behaviors. At the top, there's a condition setup with the following fields:

- Type: Control
- Control Name: TravelDuration
- Property: Value
- Operator: is less than
- Value: 14

Below the condition, there are two buttons: "Then" (highlighted in blue) and "+ Action".

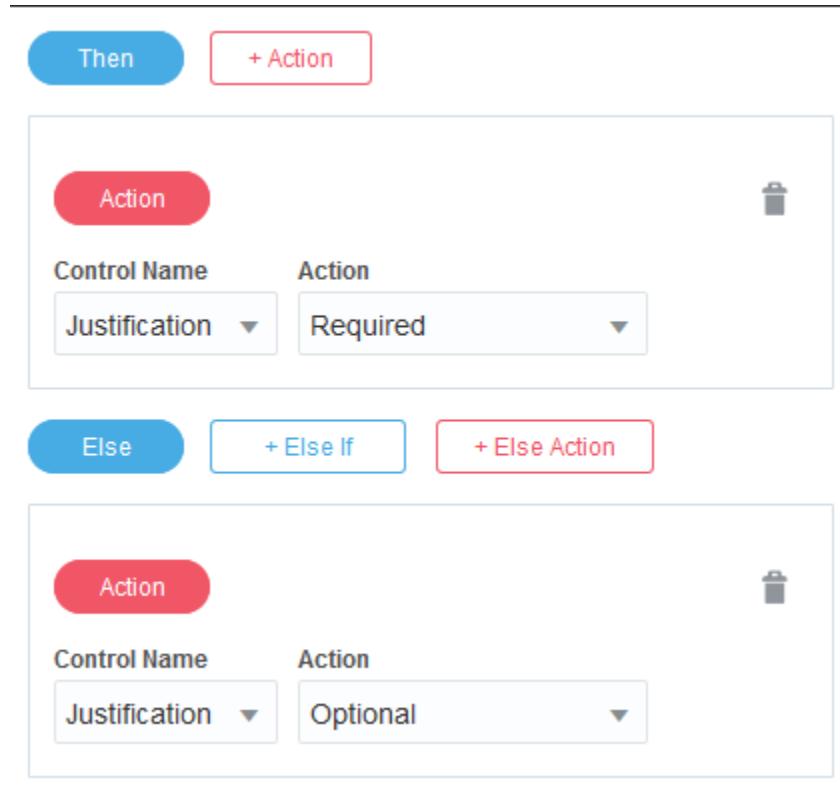
Under the "Action" heading, there's another configuration:

- Control Name: Justification
- Action: Required

Below the action, there are three buttons: "Else" (highlighted in blue), "+ Else If", and "+ Else Action".

6. Complete the Else portion of the condition by clicking **+Else Action** and completing the action.

For example, for the Cost control of a travel request form if the cost is greater than 5000 and the number of days of travel is less than 14 then the Justification field is required in the form else it's optional.



7. If needed, add and configure additional If/Then/Else conditions.

 **Note:**

In the event window, use the **Undo** or **Redo** buttons to remove or restore recent changes to conditions. Use the delete icon to delete a condition. Click **Cancel** to exit the event window without saving changes.

8. After completing the event, click **OK**, then **Save**.

Specifying Functions

Use functions in event actions and conditions to perform common operations with strings, values, and arrays. For example, use a function to add two values, concatenate two strings, or sum items in table rows.

Important points about functions:

- You can specify parameters for selected functions. The parameter value can be a constant, data definition value, control value, other function, or connector data value.
- Some functions support selecting arrays of data or repeatable controls.
- You can nest functions, such as concatenate multiple strings or the results of multiple rows. You might select a concatenate function and concatenate a data value, connector data value, or control value with another value such as a constant.

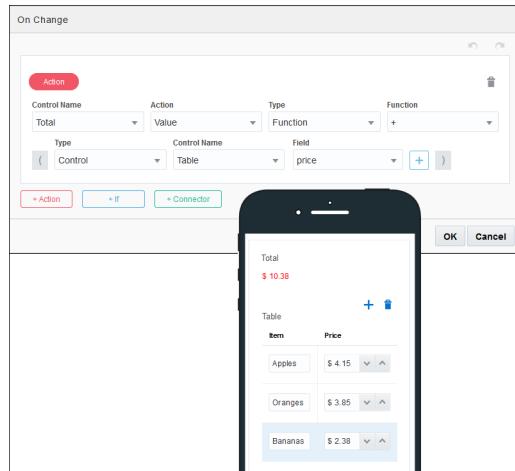
Choose from the following mathematical, aggregation, and text functions.

Function Category	Function Name	Parameters	Description
Other	Create UUID	None	Generates a universally unique identifier.
Date	Current Date	None	Generates the date in yyyy-mm-dd format.
Date	Current Time	None	Generates the time in 24 hour T00:00:00 format (for example, T23:59:59).
Date	Current Date Time	None	Generates the date/time in yyyy-mm-dd T00:00:00 format.
Math	Sum (+)	(Number, Number)	Adds two numbers.
Math	Summation	([Number])	Adds arrays of numbers.
Math	Subtract (-)	(Number, Number)	Subtracts numbers (for example, 10–5).
Math	Multiply (*)	([Number])	Multiplies numbers. For example, multiply all values in a column.
Math	Divide (/)	(Number, Number)	Divides numbers and includes the decimal portion, up to 10 decimal places (for example, 4/3=1.3333333333).
Math	Integer Division	(Number, Number)	Divides numbers and truncates the result (for example, 5/2=2, -5/2=-2).
Math	Modulo (%)	(Number, Number)	Finds the remainder after division of one number by another.
Array	Min	([Number])	Finds the minimum value in an array.
Array	Max	([Number])	Finds the maximum value in an array.
Array	Count	([Any])	Finds the count value in an array.
Array	Avg	([Number])	Finds the average value in an array.
Text	Concat	([String])	Joins a text string.
Text	Split	(String, String)	Splits a string into an array using the second parameter as a separator. For example, you can split a series of numbers in a text field into a checklist.
Text	Join	([String], String)	Joins an array into a string using the second parameter as a separator. For example, you can fetch values of all rows within a table column and create a series.
Text	Trim	(String)	Removes leading or trailing spaces.

Function Category	Function Name	Parameters	Description
Text	Replace	(String, String, String)	<p>Replaces a text string. Uses three parameters, where:</p> <ul style="list-style-type: none"> • The first string is the original text • The second string is the text to be replaced • The third text is the replacement <p>For example, REPLACE("Hello World!", "Hello", "World") returns "World World!"</p>

To use a function:

1. In the web forms editor, select a control on the canvas. Note that you can also apply events and functions to presentations, as described in [Configuring Events](#).
2. On the Properties pane, scroll down to the **Events** field. Click **Add**  to add an event and select its type in the drop-down field (for example, On Change).
3. Click **Edit**  next to the event you just added.
4. In the event window, add an action or condition by clicking the **+Action** or **+If** button.
5. Configure the action or condition to use a function.
 - For an action, select a control on which to apply the function in the **Control Name** field. In the **Action** field that displays, select **Value** in the **Action** field. In the **Type** field that displays, select **Function**, then choose a function from the functions listed by category in the **Function** field.
 - For an if condition, select **Function** in the **Type** field that displays, and choose a function from the functions listed by category in the **Function** field. Complete the condition in the remaining condition fields.
6. Specify parameters within parentheses for functions that allow them.
Additional fields display within parentheses for functions that include parameters, such as math functions.
7. Click **OK** to close the event window. Click **Preview** to test the function. For example, a simple application uses a function set on the Value field to sum values each time a user enters or changes a value in the table, and displays the calculated value in the Total field. If the total exceeds a set amount (constant), a style action displays the total in red letters.



Executing REST Connector Calls in Events

Executing a REST call in an event enables you to store the call's response and use it in an event action or condition.

Examples

- Configure a web form that prompts users to enter a zip code. Add an event that calls a weather site to query weather values. Store the response data, then add actions to the event that display temperature-related values.
- Configure a web form that prompts users to enter a company name and click a **Get Quote** button. Add an event to the button that calls a stock service site to query stock values for the specified company. Store the response data, then add event actions that display read only stock value fields.

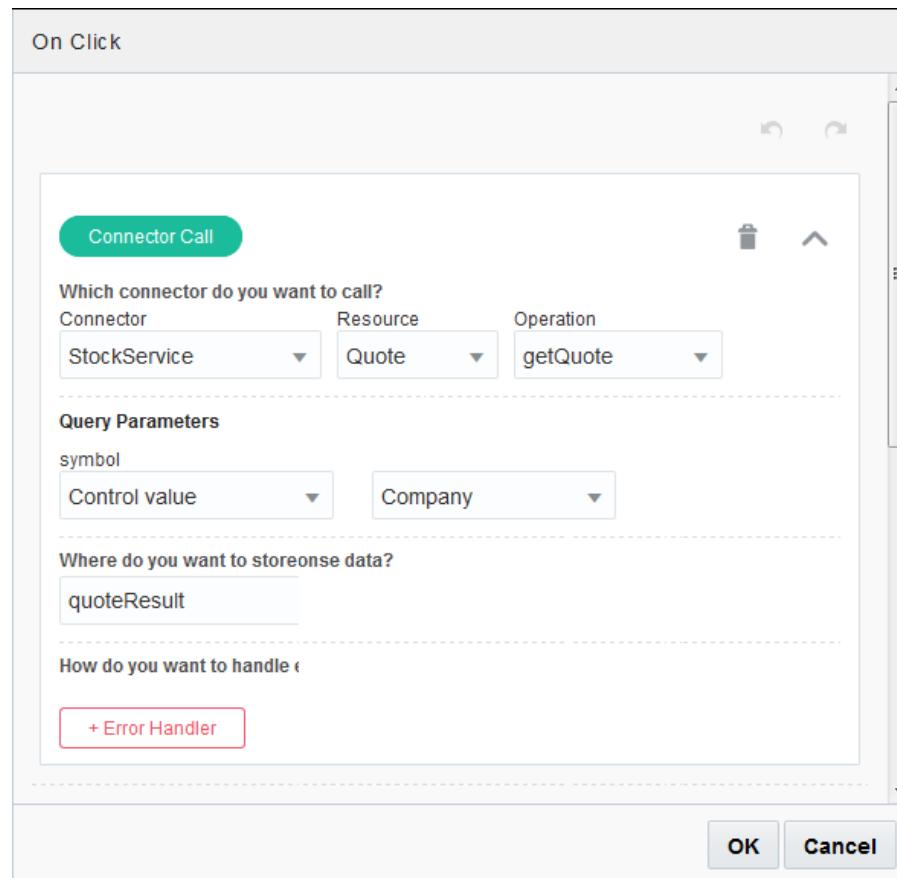
Before configuring a REST connector call, [create the REST connector](#) you want the event to call.

To add and configure the connector call:

- In the web form editor, select a control, add an event, and select its type.

For example, include an On Change event for a text input control or an On Click event for a button control. See [Configuring Events](#).

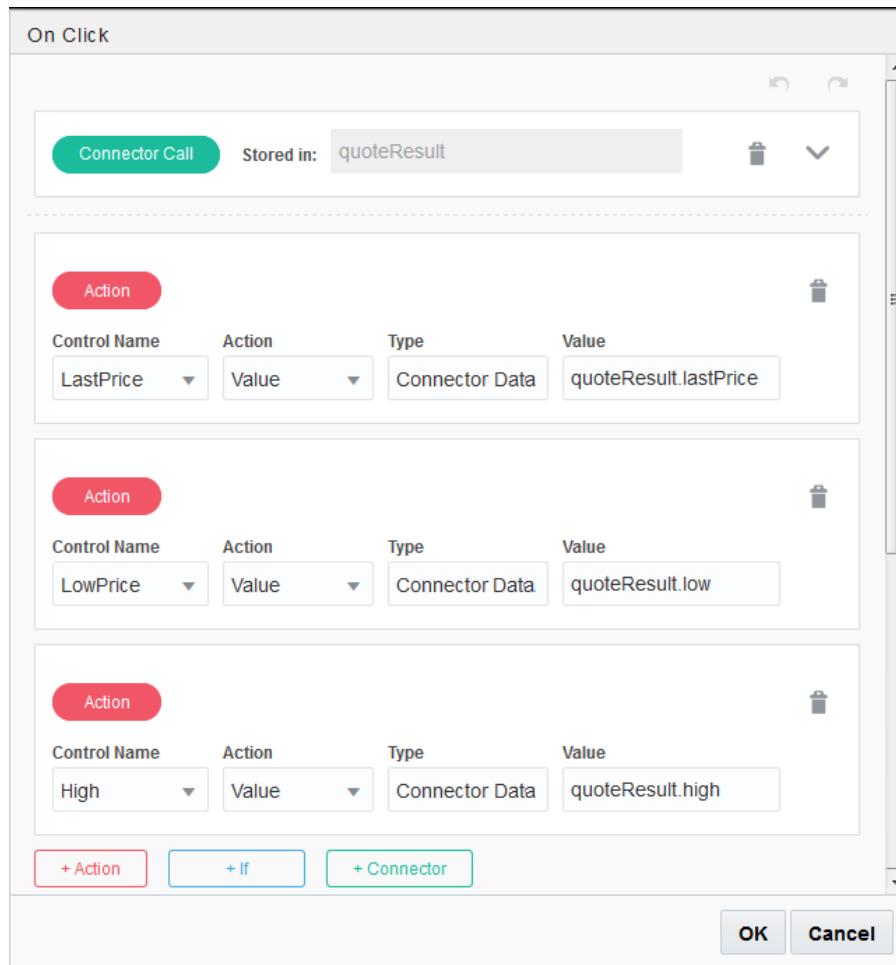
2. Click the event's **Edit** icon to display the **Events** window.
3. Click **+Connector** to add the connector call.
4. Configure the connector call.
 - a. In the **Connector** field, select a REST connector from those defined for the application.
 - b. In the **Resource** and **Operation** fields, select the resource to access and operation to perform as configured in the REST connection.
 - c. Complete connector call parameters (query, header, and template) that display based on parameters being sent in the connector call. For example, the illustration below shows the connector contains a query parameter called `symbol`.
 - d. Enter a variable name in which to store the response data (for example, `quoteResult` or `weatherInfo`).



5. Optionally, configure error handling. You can set actions to address all errors or a selected error detected in the call.
 - a. Click **+Error Handler** to add an error handler.
 - b. In the **Run on Error** field, select an error (or all errors) to detect.
 - c. In the **Control Name** field, select the control on which to identify the error. In the **Action** field, select the control or style action to take.

6. Collapse the connector call.
7. Below the connector call, configure actions, conditions, or additional connectors, and click **OK**.

A common use case is to add an action to display response data from the connector call. Click **+Action**, select a control, specify **Value** as its action, **Connector Data** as its type, and the response variable data to display in the **Value** field.



Note:

If the REST response consists of a complex data object, you cannot map the data object directly into a control (such as Input Text or Text Area) through an event. Instead, you must specify which details of the complex data object should be mapped to the control, along with an appropriate function such as **Concat** (if required), in the event window. For example in a REST response, if a complex data object called **name** contains two elements, **firstname** and **lastname**, you must specify which one of these elements should be populated into the control. If you require both of these elements to be populated, use a function such as **Concat** with a suitable separator in the event window.

Populating Controls Using REST Calls

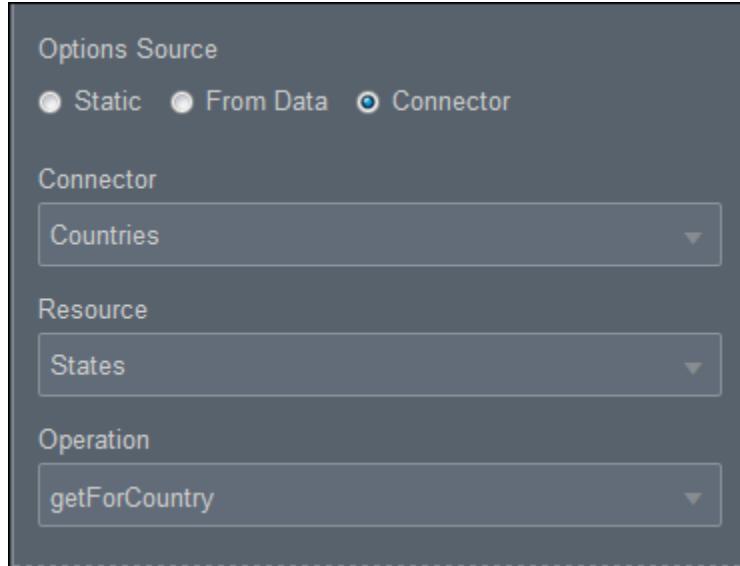
Dynamically populate controls such as drop-down select, check list, radio button, table, and repeatable section controls with data using REST connectors.

1. Drag and drop a drop-down select, a check list, a radio button, a table, or a repeatable section control onto the canvas to populate it with data using a REST call.
2. Select the control and optionally, on the **General** tab in the Properties pane, edit the name, binding, label, and other fields.
3. For a drop-down select, check list, or radio button control, in the **Options Source** field, select **Connector** and then configure the fields under the **Options Source** field to define the values based on a REST call. For a repeatable section or table control, select the **Use Data From Connector** field and configure the fields under this field.
 - a. **Connector:** Select a REST connector from the defined list of connectors for the application.

A REST connector lets you fetch data and perform tasks based on that data. See [Creating REST and Web Service Connectors](#).

- b. **Resource:** Select a resource from which data should be fetched.
- c. **Operation:** Specify an operation to call.

An operation indicates the task you want to perform. For example, a user may use an operation to fetch all the countries or to fetch a list of states based on a country code.

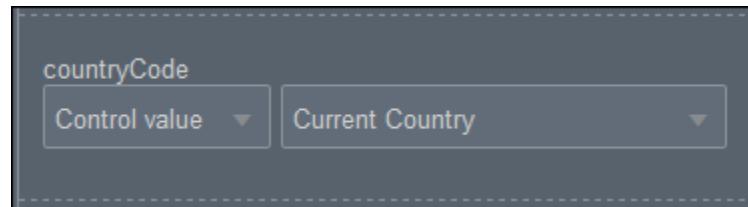


4. Based on the call, the system will display a list of required parameters (*payload values*) and a response below the **Options Source** field or the **Use Data From Connector** field.
 - a. Payload values: Specify the information to pass to the REST connector (parameters). You can specify query, header, or template parameters.

If you selected **Text**, enter the parameter information. Or, if you selected **Control value**, select a control value from the available options.

 **Note:**

Text parameters are secure and remain on the server.



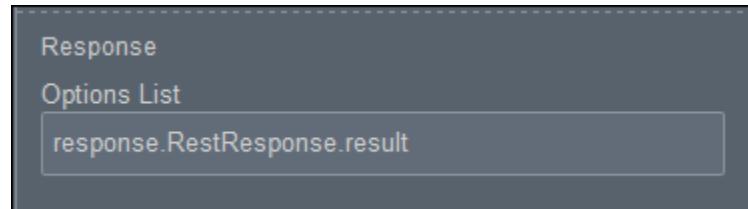
- b. **Response:** Define how the response will be mapped to the control properties:

- **Options List:** Define the mapping by specifying an attribute list from the response that contains the items to display as the options in the control.
- If the list is a simple type of list such as a list of strings or numbers, then no label and value mappings are needed. The value of the item in the list will be used as both the label and the value of the option.

If the list consists of complex elements, then you need to specify a mapping using the **Label Binding** and **Value Binding** fields to identify the label and value.

 **Note:**

- For a table control, define how the response will be mapped to each column in the table. For a repeatable section, define which property from the response will be mapped to which control inside the repeatable section.



5. You may also use the **Skip Upon Load** property to determine when the connector data loads into the control.
- Deselect the checkbox (default state) to allow connector data to be populated into the control when the form loads.
 - Select the checkbox to prevent connector data being populated into the control when the form loads. When you use this option, you must explicitly execute a connector refresh for the data to load into the control.

6. Optionally, in the **Events** field, configure the events for the control. See [Adding Dynamic Behaviors to a Form](#).

Test the dynamic fetching of data by testing the application. And, verify if the mappings you defined work correctly as expected.

Linking and Refreshing List of Value Fields

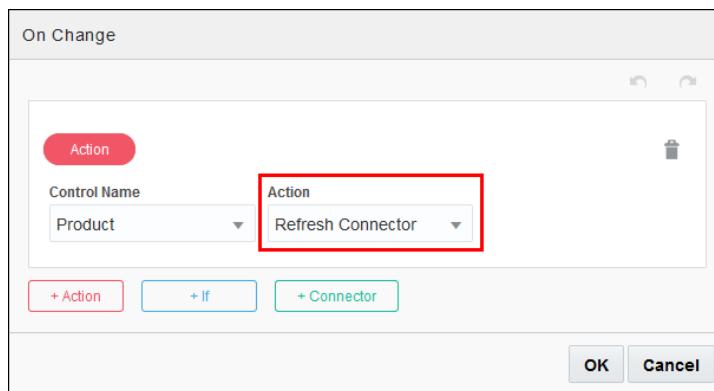
Use the Refresh Connector control action in an event so that each time an end user makes a selection in a LOV field, REST call data is refreshed and reflected in another LOV field. For example, in an order application form, after a user selects from a Category drop-down field, the list of products for that category only displays in a Product drop-down field.

To refresh an LOV field:

1. In a web form, configure two LOV controls (select, checklist, or checkbox) to use the same REST connector to populate and return values such as categories and products. See [Populating Controls Using REST Calls](#).

In this LOV use case, the Product control has query parameters set to search for the Category control's value.

2. Select the first control (for example, Category field). On the Properties pane, scroll down to the **Events** field and add or edit an event. For example, add an **On Change** event and click **Edit**  to edit the event.
3. In the event window, add an action by clicking the **+Action** button.
4. In the **Action** field that displays, select **Refresh Connector**. This setting means that each time users make a selection in the Category field, the Product field's REST call is refreshed to return product values for the selected category.



5. Click **OK** to close the event window. Click **Preview** to test the function.

Reusing Forms

You can reuse forms from the Forms Palette and use them either as reference or as copy in the forms that you build in Process. When you reuse a form as a reference, you can't change the properties or layout of the controls inside the form. When you reuse a form as a copy, all the controls inside the reused form get converted into

single controls. You can modify and update these controls like other controls in your web form.

To reuse forms:

1. Go to the Forms Palette. You can see the forms loaded into it.

If the form that you want to reuse isn't available in the Forms Palette, search for it. Click the search icon in the Forms Palette and type the form's name in the search field. If no forms in your application match the search, you will get a message stating this. Otherwise, the form gets loaded into the Forms Palette.

2. From the Forms Palette, drag and drop the form onto the canvas.

You can drop the form onto any part of the canvas. The layout of the reused form is preserved but depending on where you drop the form some panels may get added to it to maintain the form's layout.

3. By default, use the dropped form as a reference.

The reused form becomes a single control in the canvas and you can't change the properties of the controls inside the reused form.

- If **Auto Binding** is enabled, then a new type will be created in the form data definition. The data definition structure will be same as that of the reused form but it will be bound to the form that is built by you.
- If **Auto Binding** is disabled, then a warning icon will be displayed and it will indicate that the control is unbound. Define binding for the referenced form in the **Binding** field under **General** tab of the Properties pane. See [Binding Form Data with Controls](#).

4. Optionally, click **Detach** on top of the referenced form.

This will convert all the controls of the referenced form into single controls. You can modify and update these controls like other controls in your form.

This can be helpful when you want only certain parts and don't want some parts of a reused form. You can delete the unwanted controls and reuse only the controls that you want in building your web form.

Note:

- When you detach an embedded (referenced) form on another form's UI, the controls corresponding to the embedded form appear as separate entities on the UI, but the data attributes related to these controls continue to be mapped under the data attribute of the embedded form. However, if you add a new control to the detached embedded form's section, the new control and its data attribute are treated as a part of the referencing form.
- Additionally, once an embedded form is detached on another form's UI, new controls added at the source of the embedded form aren't reflected on the referencing form's UI. However, the data attribute of the embedded form reflects this change in the referencing form's data definition section. The referencing form and its associated process task use this updated data attribute.

Creating Basic Forms

In design time, you create basic forms to interact with end users. As part of creating a basic form, add its controls, configure its data, and define form behavior with basic form rules.

 **Note:**

Oracle Process Cloud Service provides two form editors: the web forms editor (recommended) described in [Creating Web Forms](#), and the basic forms designer described below.

Topics

- [About Basic Forms](#)
- [Creating and Editing Basic Forms](#)
- [Saving Basic Form Data](#)
- [Creating Basic Form Rules](#)

About Basic Forms

Basic forms define the user interface that allow end users to interact with your business processes. Oracle Process Cloud Service Composer provides an editor for creating basic 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.

Basic forms are based on standards, including XHTML, CSS, and JavaScript, which ensures compatibility across multiple platforms and browsers. At its core, a basic form is an XHTML file. However, the basic form designer allows you to create and design a basic form without interacting directly with the XHTML code. After creating a new form, you can use the basic form designer to customize the appearance and behavior of the form.

You can add the following to a basic form:

- **Form controls:** Components that you can add directly to a form.

Form controls define the graphical elements of a basic form and their layout. They also display data to users and receive data input.

When you add, arrange, or remove a form control from a basic form, the basic form designer automatically updates the underlying XHTML code.

Want to learn more about the types of form controls supported? See [Basic Form](#) and [Basic Form Control Property Reference](#).

- **Form rules:** Pieces of JavaScript code that define the behavior of a basic form or basic form controls.

Use the form rules editor to create and edit form rules. See [Creating Basic Form Rules](#).

Form-First and Data-First Design

There are three use cases for creating basic forms in Oracle Process Cloud Service:

- **Form-First:**

In this use case, you create a basic form first before any data elements are defined. This allows a business user to define the required user interface using Oracle Process Cloud Service Composer. In this use case, user-interface elements are created first without considering the required underlying data model.

After the basic form is created, Oracle Process Cloud Service automatically generates the schema that defines the data required by the basic form. This data schema is based on the basic form controls added to the basic form. When you assign the basic form to a human task, this schema is automatically used to define the human task data structure.

See [Typical Workflow Using Form-First Design](#).

- **Data-First:**

In this use case, you create a basic form based on data sources or business objects. Oracle Process Cloud Service generates data sources or business objects based on the data structure. You can add basic form controls to a basic form based on these data sources or business objects. One advantage of the data-first model is that it allows you to reuse a schema across multiple forms.

After adding form controls from a data source, use the basic form designer to rearrange them within the form to define the necessary look and feel of the user interface.

See [Typical Workflow for Using Data-First Design](#).

- **Composite of Data-First and Form-First**

In this use case, you create some portion of a basic form from scratch and generate other portions from an existing data sources or business objects.

See [Typical Workflow for Using Data-First Design](#). This workflow explains how you can generate basic form controls from business objects and then add additional controls using the designer palette.

Typical Workflow Using Form-First Design

You can use Oracle Process Cloud Service Composer to design a basic form that isn't based on pre-existing data.

This table describes the typical workflow for creating a basic form directly from the Application Home tab using form-first design methodology. After creating the basic form, you use the basic form designer to rearrange the form controls as required.

Task	Description
Create the basic form.	From the Application Home tab, you create a form with no previously defined data.

Task	Description
Add controls to a basic form.	After creating a new basic form, you can add input fields , add selection control fields , add group controls , and add other controls as necessary to define how users interact with your application.
Edit the basic form and basic form control properties.	You can edit the properties of your basic form and its controls to determine their behavior and appearance.
Add form rules to a basic form.	You can add form rules to your basic form to further customize and control its behavior.
Test your basic form.	You can use the basic form designer to preview how your basic form appears to end users.
Save your basic form.	To save a basic form, you must save the entire application.
Assign your basic form to a human task.	After completing your basic form, you can assign it to a human task . When you assign the basic form to a human task, Composer automatically generates the payload information based on the controls you added to the basic form. If the human task already has an existing payload defined, it's replaced by a new payload defined by the basic form.

Typical Workflow for Using Data-First Design

You can use Oracle Process Cloud Service Composer to design a basic form that is based on pre-existing data.

This table describes the typical workflow for creating a basic form from the human task implementation pane using data-first design methodology. After creating the basic form, you use the basic form designer to add and rearrange the data elements as required.

Task	Description
Create the basic form.	Create a new basic form from the process editor page using the human task General Properties tab.
Generate basic form controls from business objects data elements.	After creating a basic form based on a business object , you can use the business object's data elements to generate the corresponding basic form controls. The business object and data elements appears in the Data Sources pane on the left side of the basic form designer.
Add additional basic form controls.	You can add additional form controls as necessary to define how users interact with your application.
Edit the basic form and basic form control properties.	You can edit the properties of your basic form and its controls to determine their behavior and appearance.
Add form rules to a basic form.	You can add form rules to your basic form to further customize and control its behavior.
Test your basic form.	You can use the basic form designer to preview how your basic form appears to end users.
Save your basic form.	To save a basic form, you must save the entire application.

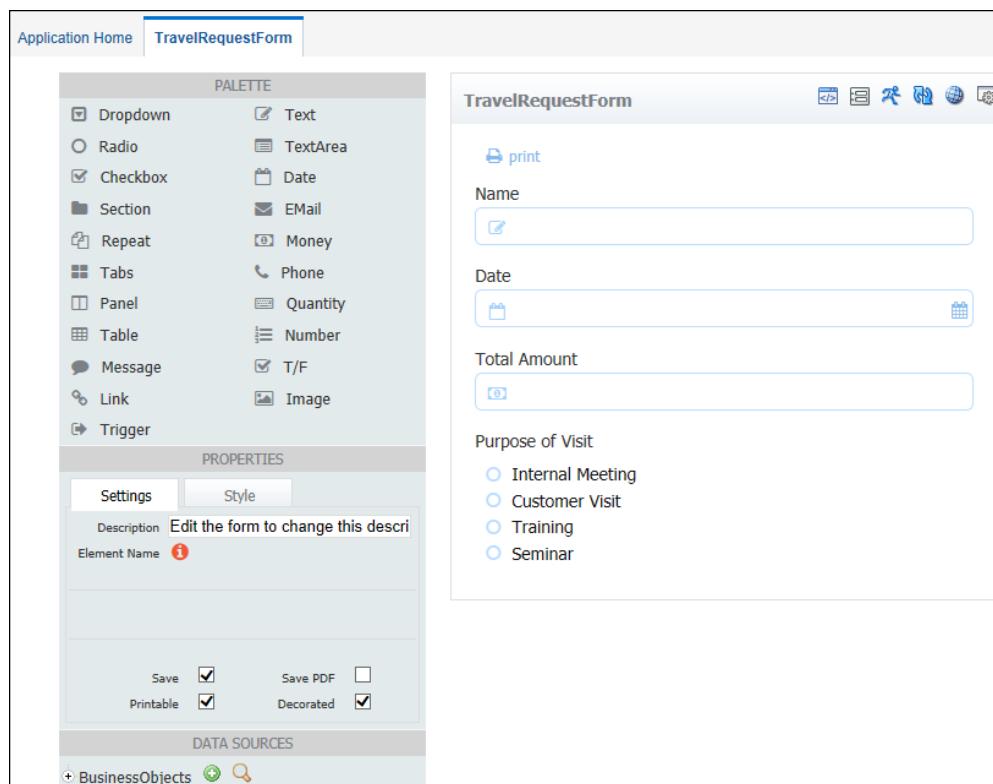
What You Can Do Using the Basic Forms Designer

Basic forms define the user interface of your process-based application.

Using the basic forms designer, you can:

- Add basic form controls to a basic form.
- Customize the appearance and layout of a basic form.
- Create form rules that define the behavior of the controls on a basic form.

Here's an example of the basic forms designer.



When you open a basic form, the basic forms designer appears as a tabbed pane. The basic forms designer is divided into the following areas:

- Basic forms component palette
- Basic forms toolbar
- Forms canvas
- Properties editor
- Data sources

Note:

Data sources are only available in basic forms created through the *data-first* use case. See [Typical Workflow for Using Data-First Design](#).

About the Basic Forms Component Palette

The component palette contains the controls you can add to a form to define how your users interact with your business application. The component palette appears on the left side of the basic forms designer. You can add a form control to a basic form by dragging the control from the palette to the form canvas.

You can add controls to a basic form by dragging and dropping from the form component palette to the forms canvas. You can drag and drop above, below, to the left, and to the right of most controls. If you're not dragging your control over a group control, your control is placed at the top of your form.

As you drag a control, the position cursor changes to indicate its status.

Position Cursor Icons	Description
Do Not Drop	The control you're dragging isn't over a valid area yet.
Drop	Your cursor is over an area on the canvas where dropping a control is allowed.
Drop Over	Your control will be placed above the control over which you're hovering. You'll see this icon when you drag the control over the top half of a valid area.
Drop Under	Your control will be placed below the control over which you're hovering. You'll see this icon when you drag the control over the bottom half of a valid area.
Drop Right	Your control will be placed to the right of the control over which you're hovering if space is available. You'll see this icon when you drag the control over the right half of another control.
Drop Left	Your control will be placed to the left of the control over which you're hovering. You'll see this icon when you drag the control over the left half of another control.

You must consider these important rules when you're dragging and dropping form controls from the component palette to the forms canvas:

- If you let go of a control while the **Do Not Drop** icon is showing, your control goes back to where it was—either back to the component palette or back to where it was in your form before you tried to move it.
- You can drag and drop above, below, to the left and to the right of most controls.
- You can drop controls above and below the *Section*, *Tab*, *Table*, *Repeat*, and *Panel* controls.
- Panels can be dropped to the left or right of other panes; however, you can't drop other controls in a similar manner to the left or right of panels.
- You can't drop two different controls into a repeat control—for example, you can't drag in a quantity control if your repeat control already contains an EMail control.

If you require multiple controls inside a repeat control, first drag the controls into a section, then drop the section into the repeat. After the section is inside your repeat you can't add anything else to the section; therefore, build the section first in this case. You also can't drop a pane, tab or another repeat control inside a repeat control.

Creating Multi-Column Forms

You can create forms with multiple columns.

All control widths are specified in columns. The width is selected by clicking on a grid in the Style tab.

When dropping controls from the palette on an empty canvas or above or below an existing control, the newly dropped control is 12 columns wide.

If you drag a control from the palette and drop it to the left or right of a control, the newly dropped control takes the width of the control you dropped on. The width of a control that's already in your form must be adjusted manually by the designer if the position of the control is changed.

To create a multi-column form:

Drag and drop panels from the palette and then drop additional controls inside the panels. Panels can be dropped to the left or right of other panels. A newly dropped panel dropped to the left or right of another panel takes the width of the panel it was dropped on.

Alternatively, drag and drop the first control onto the canvas, modify the width to your specification, then drag and drop additional controls to the right of the previously dropped one. The controls "wrap around" and line up in columns.

About the Basic Forms Toolbar

The basic forms designer contains a toolbar that provides access to form-related features. The toolbar contains the following buttons:



The basic forms designer toolbar contains the following elements:

Toolbar Element	Description
Edit Rules	Click to access the basic form rules editor. This button toggles between the basic forms designer and the basic form rules editor.
Edit Form	Click to access the basic forms designer. This button toggles between the basic form rules editor and the basic forms designer.
Preview	Opens the basic form in a new browser window. In this window, you can view how the basic form appears to end users. You can also test the behavior of basic form rules.
Refresh	Refreshes the external data source business object) of the form. If you make changes to the business object, you must refresh the data source of the basic form. This button is only applicable to basic forms created using data-first design. See Typical Workflow for Using Data-First Design .

Toolbar Element	Description
Manage Internationalization	<p>Opens the Internationalization editor.</p> <p>This editor lets you to localize components of the basic form and basic form controls. Only languages that have been added using the Language setting located in the Application Home tab are available. To configure language support, select the target language in the editor and add translation strings for the basic form's elements.</p>
Manage Business Objects	<p>Opens the Form Business Objects dialog box.</p> <p>This dialog box lets you select the business objects you want to use. You can find the selected business objects in the Data Sources section of the component palette.</p>
Manage Header Sets	<p>Opens the Header Sets Management window.</p> <p>Use it to define a group of keys/values mapped to a specific endpoint for use in requests performed from selected forms and managed in the Workspace Admin area.</p>

About the Forms Canvas

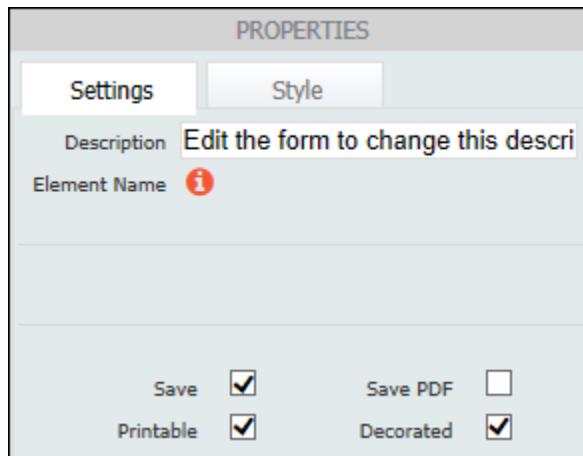
The right side of the basic forms designer is the form canvas. You can create a basic form by dragging controls from the component palette to the forms canvas. The following is an example of the forms canvas containing several controls.

The screenshot shows the 'TravelRequestForm' canvas. At the top left is a 'print' button. Below it are three text input fields labeled 'Name', 'Date', and 'Total Amount', each with a small icon to its left. At the bottom left is a 'Purpose of Visit' section containing four radio buttons: 'Internal Meeting', 'Customer Visit', 'Training', and 'Seminar'. The background of the canvas is white, and the overall interface has a clean, modern look.

About the Properties Editor

You use the properties editor to define the properties of a basic form or basic form control. When you click a control in your basic form, the **Properties** area displays the properties of the control so you can view and edit them.

The properties editor contains tabbed panes that display the properties of the basic form or basic form control grouped by function. The following is an example of the properties editor.



For procedures about editing a basic form and basic form control properties, see [Editing Basic Form Properties](#) and [Editing Basic Form Control Properties](#). Want to learn more about each property? See [Basic Form and Basic Form Control Property Reference](#).

About Data Sources

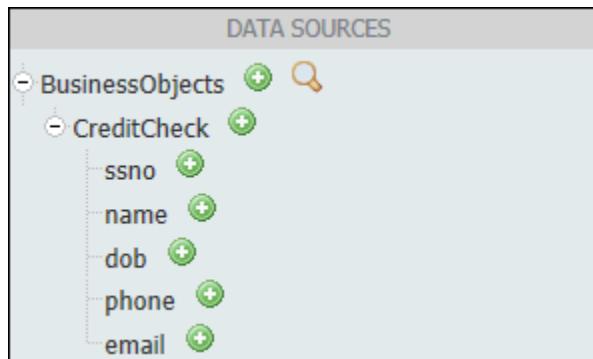
The data source pane displays the list of data elements that can be incorporated into your basic form. See [Adding a Data Source or Business Object to a Basic Form](#).

From the Data Sources pane, you can generate basic form controls based on all the data elements in the business object, or you can generate basic form controls based on specific data elements.

Note:

Data sources are only available in basic forms created based on a data-first design. See [Typical Workflow for Using Data-First Design](#).

The following example shows how the Data Sources pane appears in a basic form.



Creating and Editing Basic Forms

After creating a basic form, add its controls and data, and preview it.

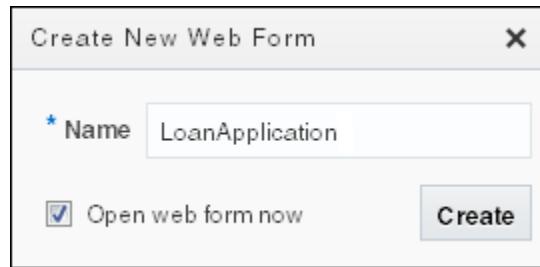
Topics

- [Creating a Basic Form from the Application Home Tab](#)
- [Creating a Basic Form from the Process Editor](#)
- [Editing Basic Form Properties](#)
- [Working with Basic Form Controls](#)
- [Editing Basic Form Control Properties](#)
- [Adding a Data Source or Business Object to a Basic Form](#)
- [Editing Basic Form Controls Generated by a Data Source or Business Object](#)
- [Showing Which Basic Form Controls Were Created by a Data Source or Business Object](#)
- [Previewing a Basic Form](#)

Creating a Basic Form from the Application Home Tab

In Composer, you can create a basic form from the Application Home tab or the process editor.

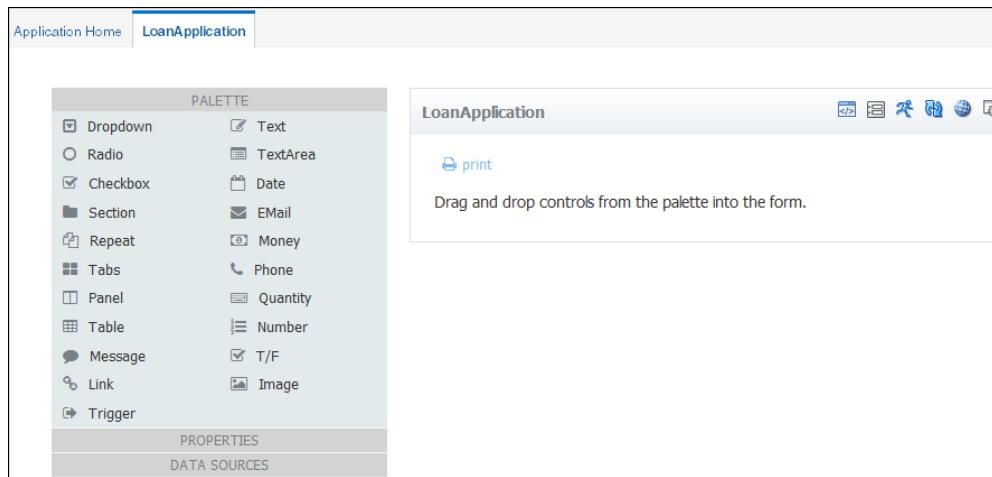
1. On the Application Home tab, select your application, then click **Basic forms**.
2. Click **New** to create a new basic form.



3. Enter a name for your new basic form, and click **Create**.

The field is populated with a default name, such as *WebForm*, *WebForm1*, and so on. You can use this name or change it. If you change the name, it mustn't contain spaces.

If you selected to open immediately, the template for your new basic form appears in the web designer.



Want to learn more about how to design your basic form? See [What You Can Do in the Basic Forms Designer](#).

Creating a Basic Form from the Process Editor

Composer provides the required tools that lets you create and design a basic form from the process editors' human task implementation tab.

To create a basic form from the process editor:

1. Create your business process.

Want to learn more about creating business processes? See [Developing Structured Processes](#).

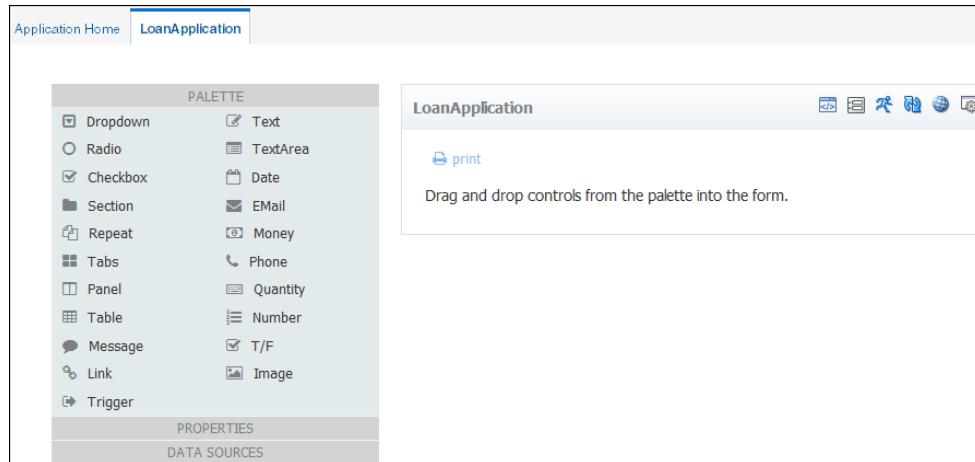
2. Add a human task to your business process.
3. Create the business objects you want to use as the base for the new basic form.
Want to learn more about creating business objects? See [Managing Application Data](#).
4. In the process editor, click the human task, select the **Menu** icon, and then select **Open Properties**.

The Properties pane opens at the bottom of the window. The General tab is selected by default.

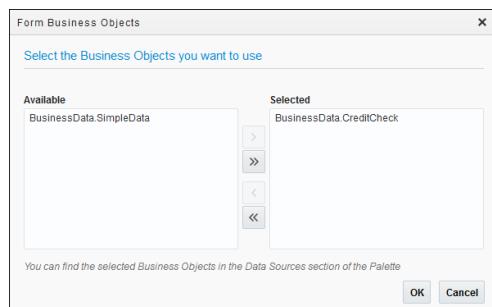
5. Select **Basic Form**, then click **Add**.
6. Enter the name of the basic form, and click **Create**.

The field is populated with a default name, such as *WebForm*, *WebForm1*, and so on. You can use this name or change. If you change the name, it mustn't contain any spaces.

If you selected to open immediately, the template for your new basic form appears in the web designer.



7. Click **Manage Business Objects** to open the Form Business Objects dialog.



8. Select the business objects you want to use.
9. Click **OK**.

You can now use the business objects data elements to generate the corresponding basic form controls. The data elements appears in the Data Sources pane.
Want to learn more about how to add the data elements to the basic form? See [Adding a Data Source or Business Object to a Basic Form](#).

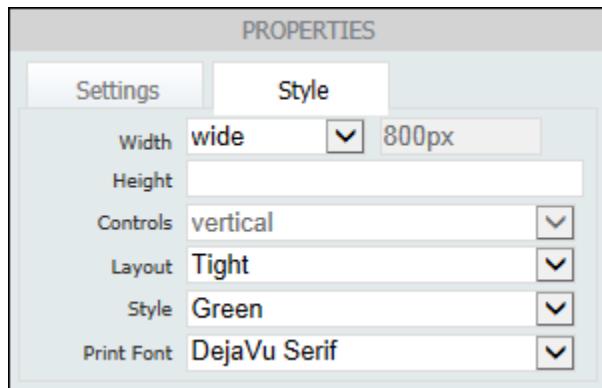
Editing Basic Form Properties

You can configure the general settings and style for a basic form.

To edit basic form properties:

1. Open the basic form and click the basic form header at the top of the form canvas.

The properties tabs for the basic form appear in the Properties pane.



2. Edit the properties for the basic form in the Properties pane.

See [Basic Form and Basic Form Control Property Reference](#).

Working with Basic Form Controls

Drag and drop controls from the palette to your basic form and configure them.

Topics

- [Adding Input Control Fields](#)
- [Adding Selection Control Fields](#)
- [Adding Group Controls](#)
- [Adding Other Controls](#)

Adding Input Control Fields

Input controls let users enter data (text, dates, numbers, and so on) into a basic form. Input controls automatically prevent users from entering the wrong data types. For example, if a user enters letters into a number control, the form displays an error message. This validation happens automatically.

To validate the content in each input control, Oracle Process Cloud Service assigns a default XML schema data type to each 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

Text

The Text control lets users enter text and is primarily intended for short, one-line entries.

TextArea

The TextArea control lets users 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.

Note:

If the value entered for the `# of Rows` is 3 the TextArea control gets populated with 4 rows. If you need exactly 3 rows, you must set the `# of Rows` to 2.

In HTML there is no property for setting the maximum length on a TextArea control. Therefore, this control doesn't have the `maxlength` property like the text control. If you must 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 [Creating Basic Form Rules](#).

Date, Time, Date/Time

The Date control lets users enter information as a date, time or date plus time control. To use Date, Time, and Date/Time controls, add a Date control to your form and use the Control Type property to select from date, time or date/time. The control is configured accordingly.

The Date Control

Use the Date control type to display the date input field with a calendar icon.

The default Date Format is Automatic. When Automatic is selected, both entry and display formatting of dates are locale specific. You still have the option to specify a particular date format that is independent of the locale by using the format field, but this isn't recommended.

If a user clicks the calendar to select the date, the date selected is displayed in the default mm-dd-yyyy format, for example 09-01-2014. Users can manually enter dates in any format, but they're then reformatted to mm-dd-yyyy.

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 (07-26-2014)
- mm/dd/yyyy (07/26/2014)
- mm.dd.yyyy (07.26.2014)
- dd-mm-yyyy (26-07-2014)
- dd/mm/yyyy (26/07/2014)

- dd.mm.yyyy (26.07.2014)
- yyyy-mm-dd (2014-07-26)
- yyyy/mm/dd (2014/07/26)
- yyyy.mm.dd (2014.07.26)

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>2014-03-06</Order Date>
```

Dates are adjusted to be within 80 years before or 20 years after if the date is expressed with a two digit year. For example, using a pattern of "mm/dd/yy" and the current date/time of Jan 1, 1997, the string "01/11/14" would be interpreted as Jan 11, 2014 while the string "05/04/64" would be interpreted as May 4, 1964. During parsing, only year strings consisting of exactly two digits, will be parsed into the default century. Any other numeric string, such as a one digit string, a three or more digit string, or a two digit string that isn't all digits, is interpreted literally. The dates "01/02/3" or "01/02/003" are parsed using the same pattern as Jan 2, 3 AD. Similarly, "01/02/-3" is parsed as Jan 2, 4 BC. Basic forms always formats for 4 digit years.

The Time Control

Use the Time control type to display only the time input field on your basic form. The Time Format property drop down menu appears, allowing 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 doesn't display the date picker.

The default Time Format is Automatic. When Automatic is selected, both entry and display formatting of times are locale specific. You still have the option to specify a particular time format that is independent of the locale by using the Format field, but this isn't recommended.

In addition to Automatic, the Time Format control has four standard formats 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.

The Date/Time Control

Use the Date/Time control to display two input fields: one for the date and another for the time.

The default Date/Time Format is Automatic. When Automatic is selected, both entry and display formatting of dates and times are locale specific. You still have the option to specify a particular Date/Time format that is independent of the locale by using the Format field, but this isn't recommended.

A property drop down menu lets you select from nine valid formats for the date of the control and four choices in military and standard time for the time. See [Dates and Times](#).

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 can't enter a time value without a date value.
- Changing the control type to Time doesn't display the date picker.
- The Time input field can't 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.

Basic form rules can be applied to the Date/Time control in all variations. Basic form rules should execute in the form time zone.

EMail

The EMail form control lets users enter a valid EMail address. The address must conform to the following syntax: <name>@<name>. <string>.

Money

The Money form control lets users enter money amounts in U.S. currency only. Users can enter commas or decimal point. If a user doesn't enter them, the form displays these symbols automatically. For example, if a user types 4000, the form displays 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 stores as many digits as the user entered but doesn't include the dollar symbol, decimal point, or commas.

Phone

The Phone form control lets users 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.

Quantity

The Quantity form control lets users 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.

Number

The Number form control lets users enter decimal numbers. Users may enter any number of digits after the decimal place.

Adding Selection Control Fields

Selection controls let users select from a list of several options.

Oracle Process Cloud Service supports the following Selection controls:

- Drop-down
- Radio
- Check box
- True or False

The basic form component palette doesn't include a Combo Box control, which is a drop-down list in which a user can type a new value or option. Combo boxes aren't part of standard XHTML. However, it's 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 that the user can enter).

Drop-down

The Drop-down control adds a drop-down list to the basic form. By default, the first choice in the drop-down list is blank. You must define the other choices by editing the Options properties of the control.

Radio

The Radio control adds mutually exclusive radio buttons. You must define the number of radio buttons and the specific choices by editing the Options properties of the control.

Check Box

The Check box control adds a set of check boxes letting the user select one or more options. Like other selection controls, you edit the Options properties of the control to define the number of check boxes and define the options available.

Booleancheckbox

The Booleancheckbox control lets a user 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 can't be changed and remain True or False.

Selecting the **Yes** check box sets the value to True in the XML document for the control data while leaving it deselected defines a False value. No value appears in the XML document.

Adding Group Controls

You can use group controls to organize your basic forms based on the types of information you require to present to your users.

Types of group controls include:

- Sections
- Optional sections
- Tabs
- Panels
- Tables
- Repeats

Sections

You can use the Sections control to create groups of controls that users can expand and collapse.

In a basic form, users can click **Expand** 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 basic 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. If you delete a Section control while designing a basic form, all controls within the section are also deleted.

Optional Sections

Optional sections can cause the validity of a form to change dynamically. For example, consider the following form:

The form consists of several input fields:

- Credit Check** section:
 - Ssno: Input field with a yellow background.
 - Name: Input field with a yellow background.
- Address** section:
 - Street: Input field with a yellow background.
 - City: Input field with a yellow background.
 - State: Input field with a yellow background.
 - Zip Code: Input field with a yellow background.
- Dob: Input field with a yellow background.
- Phone: Input field with a yellow background.
- Email: Input field with a yellow background.
- Credit Rating:
 - Excellent
 - Good
 - Poor

The required fields in the form are **Ssno**, **Name**, **Dob**, **Phone**, and **Email**. The entire **Address** section is optional. However, if the user enters an address, then the entire address is required as indicated by the yellow background when the user tabs to the **City** field. If the user enters a value in the **Street** field, the **City**, **State** and **Zip Code** fields become invalid until valid values are entered in the three newly required fields.

If the user removes the value from the **Street** field, the form validity is automatically recalculated. The **Address** section is no longer invalid because it's optional. The generated XML instance document doesn't contain an address element.

Tabs

You can use the Tabs group control to create a tabbed view.

The form includes a tabs group control:

- TravelRequestForm** title bar with icons.
- Name**: Input field.
- Date**: Input field with a calendar icon.
- Total Amount**: Input field with a currency symbol icon.
- Hotel Information** and **Airline Information**: Tabs in a group control.
- Name**: Input field.
- Street, City, State, Zip**: Input field with a location pin icon.
- Number of Nights**: Input field with a list icon.
- Purpose of Visit** section:
 - Internal Meeting
 - Customer Visit
 - Training
 - Seminar

By default, when you drag the Tab control into a basic form there are three individual tabs. To add or remove a tab, click the tab and then click **Add**  or **Delete** . Additional tabs are added to the right of the tab from which you clicked **Add**.

To rearrange the tab order, drag one tab on top of another tab. The tab you dragged moves 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 and drag the entire group to the desired location.

Panels

The Panels control creates columns within a basic form. You can add multiple columns to a form by dragging in as many panels as necessary.

By default, the width of a panel is set to 49%. When you drag two panels into your basic form, the second panel is automatically aligned with the first panel. Panels have a 1px border to make their boundaries visible. Therefore, in a two-column layout you can't make both widths 50%.

Because 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 aren't visible to users at runtime or when testing a basic form. Only the controls inside a panel are visible. These controls are organized visually according to the width of the panel.

If you delete a panel, any controls you've dragged inside it are automatically deleted.

Tables

You can use the Tables layout control to conserve space within a basic form. This control allows 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 basic form rules to calculate values, enable or disable fields, show or hide fields, and so on.

When you drag the Table control into a basic 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, which let you add and delete rows, depending on the Min/Max property values.

It's strongly recommended that column names in a table be unique.

Rearrange table columns by clicking on the arrow that appears when you click in the column heading of the column that you want to move. Columns move to the right until the last column position in the table is reached.

To remove a column, click the column header and then click “-”. To add a column, drag the appropriate control to the table and insert it between two columns. The control displays in a new column.

Table controls have the following limitations:

- Table controls can't be used in a Repeat control.
- Table cell borders aren't shown in print view.
- Table column names can't be edited from the work area. You must use the property editor.

Repeats

Repeating controls dynamically display multiple copies of a basic form control. This is useful when you want to allow users to enter multiple information of the same type; for example, a phone number. A Repeat control can dynamically display as many controls as required rather than having to explicitly add extra input controls. Repeat controls are used to repeat data elements and sections within a basic form.

You can add a Repeat control to a basic 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 basic form controls that must be repeated.

After adding a control to a Repeat control, you can select the control to view and edit its properties. Basic form controls added to a repeat control contain the following 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 don't edit the value, the control doesn't 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. Clicking on the control in the basic form designer 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 a user 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 five controls, the first three controls, in order starting from the top of the basic form, must contain data. 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 icon to add more phone numbers to the list.
</center>

When you edit the label or any of the 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.

Similar to panel controls, repeat controls are hidden when testing or using basic forms. Only the controls within the Repeat control are visible.

There are some key differences between repeat controls and the other grouping controls:

- You can't add Button, Message, or Image controls.
- You can't drag in Message, Image, Link, Tab, or Table controls.
- A Repeat control can contain only one basic form control.

You can't 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 must 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, Oracle Process Cloud Service Composer automatically generates a repeating item in the control. The schema of the Repeat control automatically enforces the minimum and maximum requirements.

Changing a Repeat Control to a Table

You can change a Repeat control to a Table control through the Control Type or Display As properties, depending on whether the Repeat control was dragged and dropped from the palette or added from schema.

When you change a Repeat control to a Table control or vice versa and there are referencing rules, it's recommended that you check the rules for the correct syntax and section names.

Adding Other Controls

In addition to Input, Selection, and Group controls, Oracle Process Cloud Service provides other basic form controls to define how users interact with a basic form.

Other types of controls include:

- Message
- Link
- Button
- Image
- Trigger

Message Control

Use the Message control to add static text on your basic form. You must provide the text in the Message property of the control. You can use the Rich Text Editor for your text; for example, you may want two lines with different font sizes or colors, the rich text editor allows you to format the text when you access the basic form. You can also use the rich text editor to change the style of the text or to add links.

To enter text in the Message Control, use one of the following editors:

1. **Editor 1:** You can add your basic text directly in the text field and that is visible in the Properties dialog box of the forms.
2. **Editor 2:** You can add your text in the Rich Text Editor.

 **Note:**

To access the Rich Text Editor, click the **Rich Text Editor** button in the Properties area of the Message Control.

3. **Editor 3:** You can use Source editor for texts with HTMLtags.

 **Note:**

To access the Source editor, click the **Source** button in the Rich Text Editor. The source editor displays and can process XHTML, but this is only for advanced users.

You can select a message type to display different pre-defined background colors, decorators, or a border from the Message Type drop-down, which is located on the Setting tab of the Property pane. These settings can override the values you have specified for BG Color and Label Color. For these values to take effect, select *None* for the Message Type.

Message types include:

- Default
- None
- Bordered
- Success
- Info
- Warning
- Error

Link Control

You can use the Link control to include a URL in your basic form. When a user clicks the link, the target URL opens in a separate browser window.

Button Control

You can use the Button control to add a button to your basic form. This control is primarily used in conjunction with basic form rules.

 **Note:**

The Button control doesn't work in repeating items.

Image Control

You can use the Image control to include an image, such as picture, logo, and so on, in your basic form. The Image control allows you to add a JPG, GIF, and PNG files or any other image type that your browser supports.

 **Note:**

The file size should be within the 5 MB limit. Files exceeding the limit will be rejected automatically.

When you drag in the Image control, a Browse button and an Upload Image button appear. Click **Browse**, go 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 must make sure that they're sized to fit within a basic form. The standard form size is 600 pixels. You can resize an uploaded image by selecting the Image control in the basic form designer, clicking on the Style tab in the Properties pane, and setting the width.

Units are px, % or em. You must include the units. For example, specifying a value of 50% resizes the image to half the width of the basic form or pane.

Trigger Control

The Trigger control adds a button to your basic form and is used in conjunction with basic form rules.

You can change the color of the Trigger button on the Setting tab of the Properties pane. You can't center the text on a Trigger or remove the decorator. The New Line property is checked by default. The Button Color can be changed by selecting one of the choices in the Button Color drop-down on the Style tab.

Editing Basic Form Control Properties

Using the basic form designer, you can edit the properties of a basic form control to configure its behavior and style.

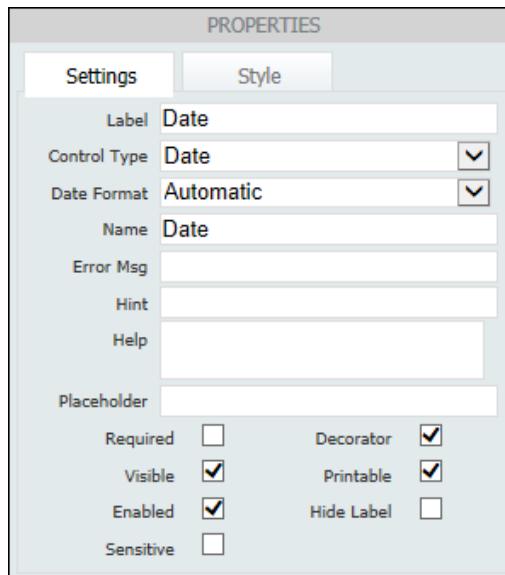
See [Basic Form Control Properties](#).

To edit form control properties:

1. Open the basic form and go to the form canvas, and then select the control.

When you select a control in the form canvas, it's highlighted.

2. In the Properties pane select either the Settings or Style tab and then edit the properties as necessary.



Adding a Data Source or Business Object to a Basic Form

After creating a basic form based on data-first design, you can automatically generate basic form controls for the data elements defined in the business object.

Basic Form Controls Generated by Business Object Data Types

Data sources enable you to create basic form controls based on existing data structures. In Composer these data structures are defined by a data source or business object. After creating a business object, you can create a new basic form based on the business object's data elements.

The following table shows the type of basic form control generated for each data type supported by business objects.

Data Type	Basic Form Control
String	Text Control
Binary	Text Control
Boolean	Boolean checkbox If checked, the value is set to <i>True</i> and if unchecked it's set to <i>False</i> 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 through the <code>Labels</code> property. However only the first option, which maps to the <i>True</i> choice is displayed. If you add more than two labels the extra labels are automatically removed.
Int / Int64	Text Control
Real / Decimal	Text Control
Interval	Text Control
Time	Time Control

Data Type	Basic Form Control
Components	<p>Section</p> <p>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.</p>

Most of the controls generated from a business object are created as text controls. However, all validation for the controls are based on the original data type `datatype`. For example an `xsd:integer` type only allows numeric values.

Schema elements that specify element restrictions are generated as selection controls. The control is created as a drop-down list if there are four or more valid choices and as a radio button if there are fewer than four valid values. You can change the display type between radio, drop-down, and check box using the `Display As` property.

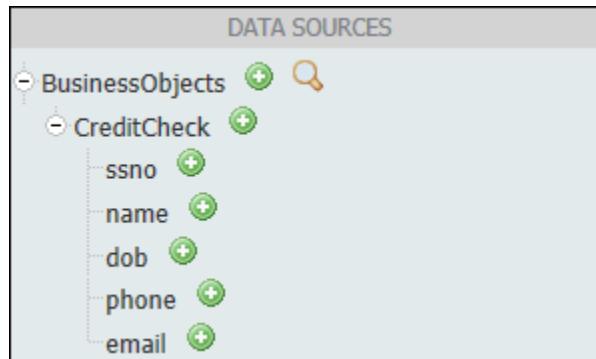
XML schemas may specify the multiplicity of an element and if it's required. For example, if an element definition specifies `minOccurs="0"`, the element isn't required. If `minOccurs="1"` then the control is required. If `minOccurs` is greater than 1 then it will be rendered in the form as a repeat control with the minimum number of items set to the `minOccurs` value. A repeat control is similar to a list containing the specified child elements.

 **Note:**

If the business object you add to the basic form is based on an XML schema that describes a cyclical or recursive structure, basic forms might generate a large number of nested controls. This can result in poor browser performance, a hang or even a crash. You should remove any schema-based business object and the generated controls from the form if the form takes a long time to render when previewed.

To generate form controls from all of the data elements or individual data elements:

1. Open the basic form you created based on data-first design.
2. Go to the Data Sources pane:



Option	Description
Generate form controls for all the data elements	<p>Click Add (+) to the right of the Business Object node in the data elements tree.</p> <p>Composer generates a basic form control for each of the data elements in the business object. The controls are added to the basic form above the currently selected control.</p> <p>Want to learn more about which basic form control is generated for each data type? See About Data Sources.</p>
Generate form controls for individual data elements	<p>a. Click the bullet to the left of the Business Object node.</p> <p>Composer displays a tree showing a list of the data elements within the business object.</p> <p>b. Select Add (+) to the left of the data element for each of the data elements that you want to add to your form.</p> <p>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're not added to a form pane.</p>

3. Edit the layout of the controls as necessary by dragging and dropping them within the basic form.

! Important:

- If you delete any data sources, you must click **Refresh**  for the changes to reflect on the basic form.
- You can't add a control generated from a data element into a repeat control.

Editing Basic Form Controls Generated by a Data Source or Business Object

After generating basic form controls from data elements you can edit the controls in the basic form designer.

You can perform the following on a basic form control after generating it from a data element:

- Edit the basic form control's name.
You can edit the name that is displayed in the basic form designer. However, the underlying XSD isn't changed.
- Create or edit basic form rules used by the basic form control.
- Set default values for the basic form control.
- Edit the layout of the basic form controls within the basic form.

You can move controls in sections or panels as required. However, you can't add or remove a generated basic form control from a repeat control.

- Edit the `Display As` property of the basic form control.
- Edit the basic form control's label.

What You Can't Edit Using the Basic Form Designer

The following changes can't be made when generating basic form controls based on data elements:

- Make edits to basic form controls that alter how the basic form or basic form controls are validated.
- Alter the number of repeats of a repeating control.
- Add additional basic form controls that require changes to the underlying data source.

Showing Which Basic Form Controls Were Created by a Data Source or Business Object

The basic forms designer lets you view which basic form controls were automatically generated from a data source or business object.

To show which controls were created from a data source or business object:

1. Open your basic form and expand the **Data Sources** pane.
2. Click **Show Form Controls** .

Each generated basic form control is highlighted in the basic form canvas.

Previewing a Basic Form

While designing a basic form, you can test its behavior directly in the basic form designer.

Each time you supply a value in a control, the form sends an asynchronous request to the server to validate the data. You don't have to complete the entire form before discovering data was entered incorrectly; try typing letters in a number control, for example, and you'll immediately see an error message. If your form includes rules that are associated with a particular control, the server applies these rules as soon as users enter data in the control.

Perhaps your form has a rule that enables the Guest Name control when you select the **Yes** option in response to the **Are You Bringing a Guest?** question. In this case, the Guest Name control is enabled immediately while you're still completing the form.

To test a basic form:

1. Open your basic form and click **Preview**  in the basic form designer toolbar.
The basic form opens in a new browser window.
2. Test your basic form by entering a date, selecting items from drop-down controls, and so on.
3. Close the preview window when you have finished your testing.

Saving Basic Form Data

Your organization may want to capture basic form data at multiple points in a process, and store it in a database or file system for archiving or auditing purposes. For example, you might capture a basic form snapshot after an end user submits a basic form and then again after a manager approves the form. The save process isn't visible to the end user.

How It Works

- Every basic form has a data object associated with it that gets created when you create a basic form.
- You use input/output [data association](#) to create a snapshot of the basic form's data object as a binary stream. Create a simple expression that uses a `Form.getWebForm` function to access the data. You can store the form's data as either a PNG (default) or a PDF file.
- You configure a service task to consume the data stream. In the service task's input data association, associate the stream data to your service method input parameter.
- Configure the service task's web service method to write the object's data to a location you select, such as a database or file system.

Form.getWebForm Function

Use the Form function in a simple expression to access the basic form's data stream:

```
Form.getWebForm(webFormDataObject,[format],[formName])
```

- `webFormDataObject` - Can be a data object or task payload object
- `format (String)` - optional - Valid values are 'PDF' or 'PNG'. Defaults to 'PNG'
- `formName (String)` – optional – Form associated with the data object. (Reserved for future use)

Function Examples

- `Form.getWebform(startWebFormDataObject)`
- `Form.getWebform(startWebFormDataObject, "PNG")`
- `Form.getWebform(taskWebFormDataObject, "PDF")`
- `Form.getWebform(startWebFormDataObject, "png", "startWebForm")`

Configuring Basic Form Snapshots

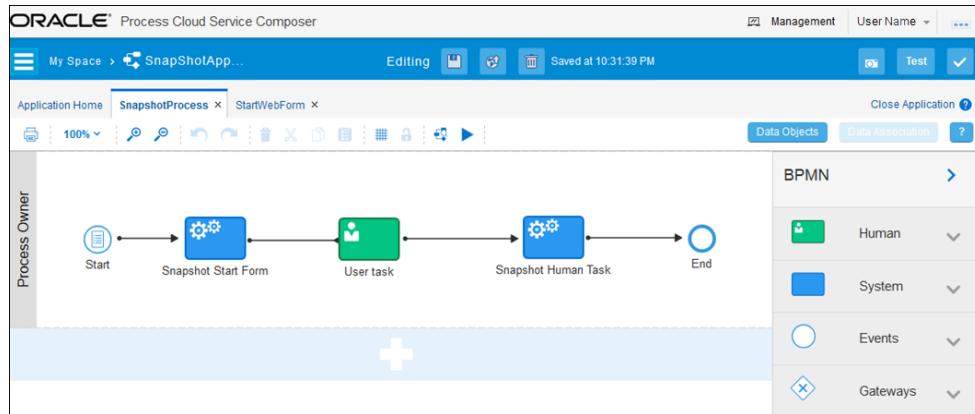
Using Oracle Process Cloud Service Composer, you can configure service tasks and capture snapshots of the basic form data and save them locally as PDF or PNG files.

To configure basic forms snapshots:

1. In the process editor, configure a process that includes a basic form and a service task.

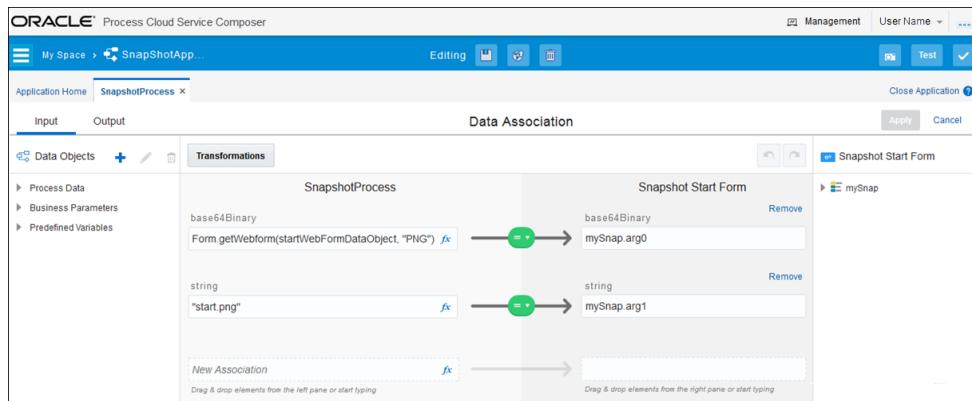
You can create snapshots for a form start event or a human task that uses a basic form. For example, in the process shown below, each service task takes a basic form snapshot:

- First, when the end user clicks Submit for the start form
- Second, when the end user clicks Approve for the human task



2. Set up an expression for the service task that performs the basic form snapshot.
 - a. Select the service task, and select **Data Associations**.
 - b. On the input side of the **Data Association editor**, click the **fx** (Expressions) icon.
 - c. In the **Expression Editor**, click the **Operators** tab, expand **Form**, select the **Get WebForm** function, and click **Insert Into Expression**. Note that you can enter Form. in the expression field to trigger an autocompletion that lists the three available formats for the function.
 - d. In the expression, insert the data object you want to snapshot, and identify an output format of PNG or PDF, and other optional parameters.
 - e. Click **Validate**.

This expression creates a binary stream of data you can associate with your service method input parameter.
3. Optionally, set up a second data association that stores a snapshot file name, which you can use to identify the file in your service method.



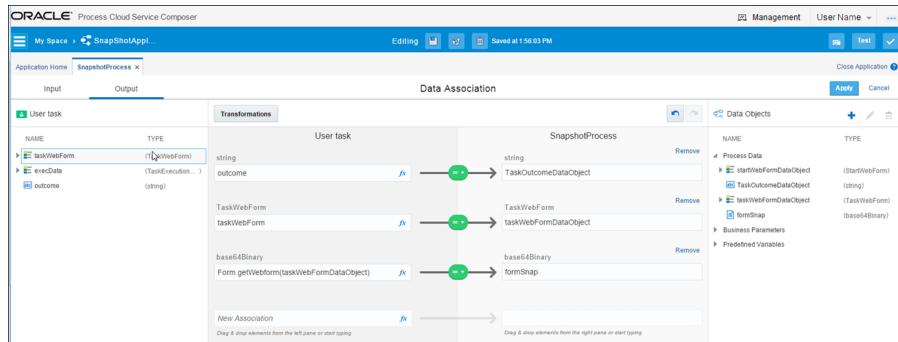
4. Optionally, store the snapshot stream in a base64Binary data object for later use as input in your service task data association.

This step enables you to store the snapshot in the data object and then save all the form snapshots at once at the end of the process.

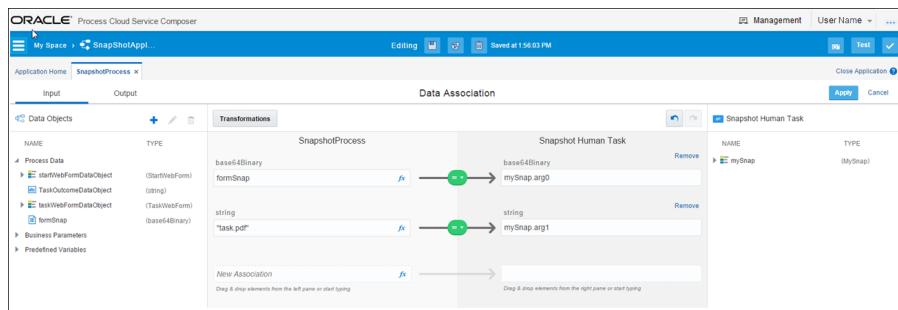
- a. Create a base64binary process [data object](#) to store the captured data stream.

The dialog box is titled 'Add Data Object'. It contains a 'Name' field with the value 'formSnap', a 'Data Type' section with 'Simple' selected and 'base64Binary' chosen from a dropdown, and a large 'Add' button at the bottom right.

- b. Populate the data object using the output data association of the start form or human task. This method is executed when the start form is submitted or the human task is acted upon.



- Use this process data object in your service task's input data association.



The basic form's data object always contains the form's latest data. If the same form is editable at multiple process stages, it's useful to store the form data at each level in a different process data object as described in step 4.

- Create a web service for the service task that calls a method to write the file.

Below is an example of the service class implementation with a method that takes the basic form data stream and stores the file in a local file system.

Example Method

```
public String mySnap(byte[] content, String name) {
try {
String fileName = "/scratch/" + name;
File file = new File(fileName);
BufferedOutputStream writer = new BufferedOutputStream(new
FileOutputStream(file));
writer.write(content);
writer.flush();
writer.close();
} catch (Exception ex) {
}
return "Done";
}
```

- Deploy and test the application, submitting and approving the form as needed. Check the location where the web service stored the form data and view the PNG or PDF files.

Creating Basic Form Rules

Form rules are pieces of JavaScript code that let you define how users interact with your basic forms. You must be familiar with the JavaScript to create and edit form rules.

Topics

- [JavaScript Syntax for Basic Form Rules](#)
- [Creating a Basic Form Rule](#)
- [Using Dynamic Content](#)
- [Using Dynamic Validation](#)
- [Using Built-In Data](#)
- [Using Built-In Methods](#)
- [Testing and Debugging a Basic Form Rule](#)

JavaScript Syntax for Basic Form Rules

Basic form rules are pieces of server-side JavaScript code that let you determine how users interact with your basic forms by defining how basic form controls are displayed and the types of data users can enter.

You can use basic form rules to define the behavior of a basic form. Typical uses of basic form rules include:

- Adding dynamic behavior to a basic form, including showing or hiding elements or enabling or displaying elements.

For example, you can add basic form rules that 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 basic 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.

For example, you can retrieve a list of values for a drop-down list from a REST service. Retrieving values from a secured service requires you to create a security key credential. See [Configuring Credentials for Web Services](#).

General Format and Guidelines

In Oracle Process Cloud Service, a basic form rule is expressed in JavaScript code. Basic form rules are saved and can be run after you save your application.

Basic form rules generally have the following form:

```
if (condition)
{
```

```

        true actions;
    }
else
{
    false actions;
}

```

You can create more advanced basic form rules primarily for the purpose of looping over repeating items. Here are some basic characteristics of JavaScript that you must be aware of when writing basic 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 don't 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 Process Cloud Service doesn't support the following JavaScript syntax:

- **switch statement**

Oracle Process Cloud Service does support 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 basic form rule:

```

if (form.load) {
    try {
        x.randomproperty = 'blah';
    } catch (e) {
        FN.value = 'got exception';
    }
}

```

However, catching an exception when a form control isn't defined isn't supported and may behave unexpectedly. This is because Oracle Process Cloud Service catches the problem while parsing the basic form rule before the JavaScript interpreter catches the error.

In the following example, the control named Color doesn't exist in the form.

```

if (form.load) {
    try {
        Color.value = "red";
    } catch(e) {
        msg1.value = "error caught: " + e;
    }
}

```

Control Name

Basic form rules often must 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 basic form rule as `FirstName.value`. `firstname.value` doesn't work.

When using a control in a basic form rule, you must ensure that the control has a unique name. If multiple controls have the same name, the run time environment can't determine which control the basic form rule refers. Form controls added to a form from the palette are usually guaranteed to have a unique name. The basic forms designer doesn't 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 name is automatically made unique. If you try to edit the name such that it's no longer unique, the basic forms designer prevents you from making the change.

When a control is dropped inside a section control, it's at a different nesting level than a control dropped outside a section. Two controls, one inside a section and another outside the section, are also at different nesting levels. The basic form designer allows you to provide identical names to the controls.

If non-uniquely named controls are used in basic form rules there may be unexpected results. If you encounter errors in a form, you can edit the names of control 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.

Basic Form Rule Identifiers

Basic 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 basic form rules using the name as an identifier.

Form rule identifiers must always be of the following form:

`Name.<property>`

These basic 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 isn't applicable to sections, tabs and other controls that don't have values displayed to the user.

- **Enabled:** Set to *False* to disable a control so that a user can't change its value and set to *True* to enable it.

This isn't applicable to sections and tabs.

- **Expanded:** Set to *False* to collapse a group control (sections controls only) and set to *True* to expand a group control.
- **Selected:** Set to *True* to designate a tab as 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 is invalid if the user enters a string value that can't 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, drop-down and check box 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 to *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 through a Document URI as the form loads, its state turns to *True*.

- **itemRemoved:** Used by repeat controls. Its initial state is *False*.

When a user clicks "-" to delete a repeat item, its state turns to *True*.

- **itemIndex:** Used by repeat controls.

When an `itemAdded` or `itemRemoved` event fires, the value of `itemIndex` is set. For `itemRemoved` events `itemIndex` returns -1. For `itemAdded` events `itemIndex` gives you the index of the added item.

- **form.load:** This property is set to *True* when the form is first loading.

Its useful for setting default values through basic form rules that must be set before the user starts interacting with the form.

- **form.unload:** This property is set to *True* when the users clicks the form's Submit button.

It's useful for setting control values just prior to when the form's Doc Actions and Form Actions are executed.

Examples of identifiers used in basic form rules are:

- FirstName.value
- BillingAddress.visible
- Email[1].value
- Email[i].visible

The latter two are examples of repeating controls. Repeating controls are discussed in more detail later. Note that the case of the properties is important. FirstName.value is a valid basic form rule identifier but FirstName.Value or FirstName.vAlUe aren't.

Strings and Numbers

Because JavaScript is a loosely typed language there may be situations where you must add field values and the basic form rule performs string concatenation instead. There are several ways to tell the basic form rule to perform mathematical calculations instead of string manipulation. One simple way is by adding a *1 to the variable. id = id*1 + 1; ensures that id equals the current value plus one rather than the current value with a 1 appended. For example, if the current value is 4 and you didn't write id*1 the result may be "41" rather than 5.

You may also encounter a situation in a basic form rule in which money controls perform a string concatenation rather than addition. To correct this:

- Open the form with the basic form rule in the designer, change the money controls to text controls, then save the form.
- Open the form, change the text controls back to money controls, then save the form again.

Dates and Times

To initialize a date control, the data must be in the correct format.

Here are some examples of correct date formats:

- d1.value = "2012-01-13"; //yyyy-mm-dd
- d2.value = "01-13-2014"; //mm-dd-yyyy
- d3.value = "jan 13 2014"; //using string for month

In general, it's not recommended to rely on either the browser's locale or the server's locale when writing date, time (time of day) and date/time literals in rules.

An example date and time rule follows:

```
if (form.load) {  
    date.value = "2014-02-29";  
    timeofday.value = "13:10:02";  
    datetime.value = "2014-02-29T18:10:02Z";
```

This rule initializes a date control in a form with 02-29-2014, a time control with 1:10 PM and a date and time control with 02-29-2014 in the date portion and 1:10 PM in the time portion.

Notice the special character "Z" at the end of the date/time value. This is equivalent to an offset of UTC-00:00 and indicates that the time of 18:10:02 is given in UTC. 18:10:02 UTC is equal to 1:10 PM Eastern Standard time in UTC.

Time and date/time controls also require correct formatting and an understanding of how basic forms manage time zones. Basic forms convert times into the browser's local time zone before displaying them in the form. However, if time data isn't correctly formatted, basic forms make multiple attempts to parse the time but always displays the time with NO time zone conversion. Here are examples of correct time formats:

- `t1.value = "20:30:00";`
- `d2t2.value = "20:30:00Z";`
- `t3.value = "20:30:00-04:00";`
- `dt1.value = "2012-08-23T20:30:00";`

Writing Conditions

One of the most common conditions is a form rule that executes as soon as the user enters a value in the 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 basic form rule can be simplified. This basic form rule executes whenever a user enters a value into either the control named `firstname` or `lastname`.

```
fullname.value = firstname.value + ' ' + lastname.value;
```

Select Controls

Radio controls, drop-down lists, and check boxes are all examples of select controls.

Radio controls and drop-down lists are single select. Users can select only one item from the radio controls or in the drop-down list at a time. Thus, the `ID.value` of radios and drop-downs is similar to the other input and output controls. The value is a single item.

Check box controls are multi-select. Users can select multiple items at any given time. Thus, the `ID.value` of a check box is an array. For check boxes, a valid expression is `ID.value.length`, which returns the number of items in the value array. `ID[0].value` retrieves the first item in the list, `ID[1].value` retrieves the second item, and so on. If you have a check box control with only one check box and that check box is unchecked, then the array contains no elements. For a check box control a useful expression is `ID.value.length == 0`. Note that `ID.length` isn't a valid expression. Also, because a check box control is an array it isn't a valid expression to write `ID[0].value == .` This is because if an option in the check box control is unchecked, its value isn't an empty string. It simply doesn't exist in the array.

Initial Control State

Every control in your form has an initial default property state for the visible, expanded, value, valid, and enabled properties. However, you can change controls in the initial state in the form designer Edit tab. An initial state of a control 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 through the controls edit property pane. The Edit property contains check boxes for visible and enabled for controls on which those properties make sense like input controls.

Basic Form Rules and Repeating Controls

If you have a repeating control in a form that itself contains a repeating control, you can't apply a basic 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.

Additional Examples

See [Basic Form Rule Examples](#).

Creating a Basic Form Rule

Oracle Process Cloud Service Composer provides an editor that lets you create and edit a basic form rule.

To create a new basic form rule:

1. Open the form where you want to create a new form rule.
2. Click **Rules** located on the toolbar.
3. Click **Create New Rule**, click **Edit**, and then edit these fields:
 - **Name**: Defines the name of the basic form rule. Change this to something meaningful.
 - **Enabled**: Select to enable the basic form rule within your form. If you deselect this option, the basic form rule isn't applied to the form at runtime.
 - **Description**: Provides a description of the basic form rule. Modify the default to describe the behavior of the basic form rule and how it functions within the context of the form controls and other basic form rules within the form.
 - **Rule**: Defines the form rule.
4. Click **Edit** located on the toolbar to return to the basic forms designer.

You can test the behavior of basic form rules while testing a basic form. See [Previewing a Basic Form](#).

Using Dynamic Content

Real forms often require dynamic content in drop-down lists. Often based on the value entered into one form field, the values in other fields or options available in select controls must be dynamic.

Basic 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, drop-down, check box) controls. You can also use `http.post()`, `http.delete()`, and `http.put()` in basic form rules, although you must use URL parameters with them, as they don't all support payloads.

Here's an example that shows the syntax of the `http.get`. This basic 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`. You can then use `x` in the basic form rule as necessary. In this example, it's 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;
```

Another example is 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 drop-down list.

```
eval('x=' + http.get('http://<webhost>/test/json/getClients'));
Clients.options = x.clients;
```

Here's another example of a servlet that returns a JSON object after authenticating a user and 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 can 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>";
    }
}
```

It's important to note that the `http.get` is accessing your HTTP service through a URL. Certain characters must be encoded in order to correctly pass through an HTTP URL. You may not always be able to control the values your users enter into your form

fields. If the value may contain one of these characters, then that value must be encoded.

For example, this `http.get` contains a password parameter. Because user passwords may contain characters such as the # sign, 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 have 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");
```

Reusing Dynamic Content

Fetching dynamic content from your back end system can be the source of degrading form performance. If your form must 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. You can then later read the text control value and eval it again rather than having to call your back-end system multiple times.

For example, add a hidden text control to your form named `jsonData`. Add a single line to the basic form rule that retrieves the user data from your back-end system through the `http.get()`:

```
jsonData.value = x;
```

In other basic form rules that also require the user data, rather than calling `http.get()` to your back-end system again, add the following line to your basic form rule:

```
var val = jsonData.value;  
eval('x' = val);
```

This has the affect of setting `x` to the same string as if you had again fetched content from your back-end system.

Using Dynamic Validation

You can now perform dynamic basic form validation based on what action is being taken in Workspace.

Workspace passes the action name as a parameter and the JavaScript passes it to the basic form rule. The basic form rule decides whether the form data is valid or not based on the action.

To use dynamic validation:

1. Create a text control with the name `_action` in your basic form and make this field *hidden*.

 Note:

At runtime, the value of this field is set based on the action selected by the user.

2. Write a rule that can do validation based on the value of `_action`. For example:

```
if(_action.value === "APPROVE")
{
    email.required = true;
}
else
{
    email.required = false;
}
```

In this example, if the user clicks on **Approve**, then the form makes `email` required. If the action taken is anything other than **Approve**, then `email` isn't required.

Using Built-In Data

Oracle Process Cloud Service provides certain data to your basic form rules. This includes the Id of the person currently using your form. You retrieve this data in your basic form by calling `var user = decode(subject.id);`.

To obtain other user attributes, you can use Oracle BPM Rest Identity Service. Assume you have a text field named `UserLastName` and you want to populate it with the logged in user's last name. You can accomplish it by using the following basic form rule.

You can use a `form.load` basic 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`.

```
if(form.load)
{
    var loggedinUser = decode(subject.id);
    eval('user=' + http.get('http://host:port/bpm/api/1.0/identities/user/' +
loggedinUser));
    UserLastName.value = user.userLastName;
}
```

Using Built-In Methods

Oracle Process Cloud Service provides built-in helper methods for common functionality required by basic form rules.

The following is a 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 time zone.

This method should only be used to set the value of a Date control.

- `DateTime frevvo.currentTime(form)`: Returns the current date and time in the user's local time zone.

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()` doesn't work in a `form.load` basic form rule unless you specify a time zone on the form's URL through the `_formTz` URL parameter. This is because the form server must know the time zone in which to return the date and time. If you don't specify a `_formTz` the methods return null and the control values remain blank. For example, to specify Eastern time: `&_formTz=America/NewYork`.

Use the following methods to work with users and roles:

- `boolean isUniqueUserId (String userId, String tenantId)`: Returns *True* if this user doesn't exist in the tenant and *False* if it does.
- `boolean isUniqueRoleId (String roleId, String tenantId)`: Returns *True* if this role doesn't exist in the tenant and *False* if it does.

Working with REST Calls with Custom Headers

For basic forms that fetch data by calling external REST services that are protected by custom headers, Oracle Process Cloud Service provides custom header support using a superset of Basic Auth support.

Custom headers use tokens for authentication instead of a user name and password and aren't shared with Composer during REST calls. The header can be marked as secret or can be made visible to Composer users.

You can map multiple headers to the same REST API call. For example, two departments can create different headers and map them to the same REST API call.

How do I work with custom headers?

The following steps summarize the tasks for custom header support:

1. An administrator [adds authorization in Workspace](#).
2. An administrator [adds a custom header in Workspace](#).
3. An administrator shares the token with the developer.
4. A developer [maps the token to a basic form in Composer](#).

Managing Header Sets Used in Basic Form Requests

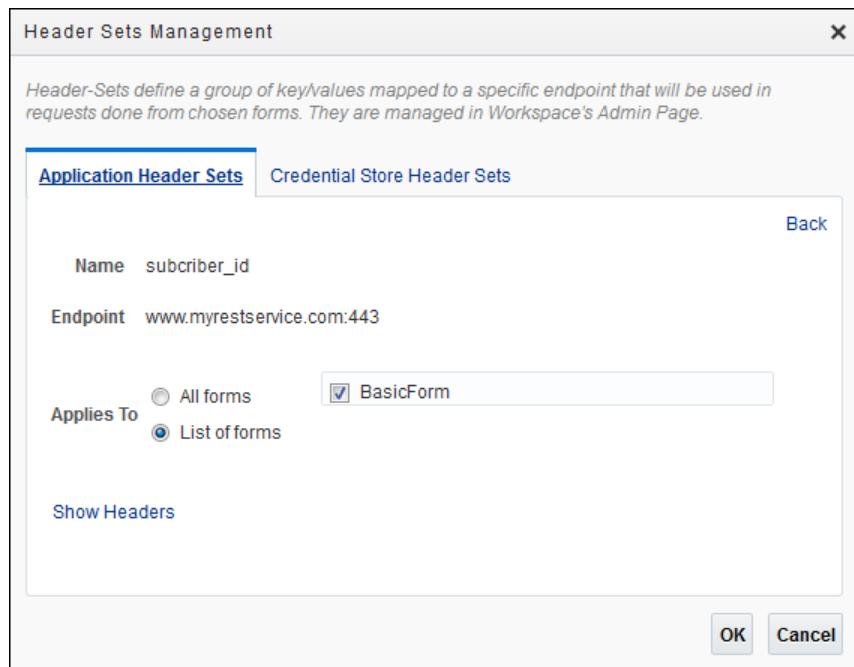
You can map the header sets to the basic form, view the available headers, and edit and delete the headers in Oracle Process Cloud Service Composer.

The Header Set Management page contains the following tabs:

- **Application Header Sets:** Displays the header sets that are used in the application. You can edit and delete the header sets.
- **Credential Store Header Sets:** Displays the header sets that are available for use in application. You can click **Show Link** to view the details of the selected header.

Mapping Custom Headers with Basic Forms

1. On the Composer Home page, click **Forms**.
2. On the Forms page, open a basic form.
3. In the basic forms designer, click **Manage Header Sets**  on the editor's toolbar.
4. On the **Credential Store Header Sets** tab of the Header Sets Management dialog box, select a header to map. You can select more than one header.
5. Click **OK**. The selected header now appears on the Application Header sets tab.
6. On the Application Header sets tab, click the **Edit** icon for the header that you want to map to the form.



7. Select one of the following options:
 - **All forms** to map the header to all forms
 - **List of forms** to map the header to one or more selected forms
8. Click **OK**. The selected header or headers are mapped to the form.

Testing and Debugging a Basic Form Rule

When you create or edit a basic form rule, Oracle Process Cloud Service determines the list of controls and properties of those controls that the basic form rule depends upon. The basic form rule is automatically executed whenever there is a state change that is relevant to the basic form rule. Form rules are also executed sequentially in a top to bottom order as they're seen in the form rules pane.

Basic form rules can trigger the execution of other basic form rules. So, if a basic 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 basic form rule R2 is triggered and executed.

A basic form rule typically refers to one or more form controls and their properties, and is executed if any of those properties change value. Basic form rules aren't fired when the page is loaded; for example, the following basic form rule is only executed when `N1.value` changes its value.

```
if (N1.value > 0 || N2.value > 0) {  
    T.value = N1.value + N2.value;  
}
```

Assume a use case where you want to show a message when 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*. You can write the basic form rule as:

```
if (!E.valid) {  
    // code to show message here.  
}
```

The previous code doesn't work as expected as E is a required field and therefore `E.valid` initial value is *False* since the field is initially empty. When the user enters an invalid Email address, `E.valid` still has the value *False* and the basic form rule can't execute since there's no state change. The following code works properly:

```
if ((E.value.length > 0) && (!E.valid)) {  
    // code to show message here.  
}
```

The basic 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 basic form rule is executed and the message is shown.

About Infinite Loops

It's easy to create basic form rules that enter a loop condition. For example, a basic form rule that updates `A.value` based on `B.value` and another basic form rule that updates `B.value` based on `A.value`. They can continually trigger each other.

Oracle Process Cloud Service prevents this from happening by setting an execution time limit for each basic form rule. Any basic form rule that takes longer than 5 seconds to execute is forcibly stopped. This is a very lengthy period of time for most computational tasks and the vast majority of basic form rules aren't impacted by this constraint. However, since Oracle Process Cloud Service is a hosted site that may be simultaneously used by numerous users, there is a time limit imposed on these computations.

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 named Address. However, if you want to use either of the Address controls in a basic form rule, you must give them unique names.

One way to tell if a basic form rule is breaking because of duplicate names is to look at the error sign in the Tomcat console. If you see a message similar to the following, 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...* is Java interpreting the controls with the same name as an array, not single controls.

About Basic Form Rule Profiling

Basic form rule execution can be profiled to determine how much time each basic form rule takes to execute. This can help you tune and improve performance in forms that use a large number of basic form rules. To turn on profiling set the `rule-debug=profile` on the URL.

If your basic form rules are executing database queries to populate dynamic select controls (drop-down, radio, check box), basic form rule profiling can help you identify if the database queries are taking too long. You may see HTTP calls taking a long time to execute.

Managing Application Data

Almost all business processes require some type of data. For example, a purchase order application requires customer name, contact information, order number, items purchased, and payment details. Before you can use data in your application, you must define how it's structured and stored.

Topics

- [About Managing Data](#)
- [Defining Data](#)
- [Associating and Manipulating Data](#)

About Managing Data

Most business applications require users to create and manipulate data. In a travel request application, for example, a user enters data related to the request which includes information about the employee, to and from destinations, and other types of data. Additionally, an application may have 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 that the Process application uses. Data is stored within a data object. Data objects are defined based on complex data types.

In Process, business types consist of data definitions referred to as business objects and business exceptions. A business object is a complex data type that groups together related data. For example, if you're creating an application that requires information about an employee, you may want to create a business object that stores 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. Create business objects from scratch using the Business Object Editor or automatically by importing an XSD. Business objects are also auto-generated when designing forms using the form-first design methodology and when importing services into an application. In these situations, business objects are referenced by forms and services and can only be deleted if the artifacts referring them are deleted. Business objects can be hierarchical - attributes can be simple or point to other business objects.

Defining Data

To use data in your application, you must first define how it's structured and stored.

Topics

- [Typical Workflow for Defining the Data Used Within a Business Process](#)
- [What You Can Do on the Business Types Page](#)

- Working with Business Objects
- Working with Data Objects
- Creating a Business Exception
- Creating an Enum Object
- Working with Business Indicators

Typical Workflow for Defining the Data Used Within a Business Process

Defining how data is stored and manipulated is part of the overall design and development of an application.

Use the following typical workflow to define the data used within your Process application. The first three tasks are required; complete the remaining tasks as needed.

Task	Description
Define the business types	Business types, such as business objects, business exceptions, and Enum objects, define the data structures used within your application.
Create data objects	Data objects store the information used within your business process. Associate each activity with your business process with the data objects it requires.
Configure data associations and transformations	Data associations define how information is passed between the flow elements in your business process.
Define expressions	Expressions manipulate the data used in your business process.
Create business indicators	Business indicators store the key performance information used in your business process.
Define input and output arguments	The flow events that you use to define your business process operations allow you to define input and output arguments. These input and output arguments define the process input and output.

What You Can Do on the Business Types Page

The Business Types page provides the tools required to create and manage business types (business objects, enumeration (enum) objects, business exceptions), and also separately manage the life cycle of XML Schema Definition (XSD) files. You can use the business types in any process created for the selected application.

The Business Types page is divided into two sub views:

- Business Types
- Schema Files

Use the view option to alternate between views.

 Business Types Schema Files

Business Types View

Use the Business Types view to perform the following tasks:

- [Create business objects.](#)
- [Create business exceptions.](#)
- [Create enum objects.](#)
- [Import business objects.](#)
- Click **Views** to view the business types in Grid or List format. Grid view is the default.

Each business type displays: its name, type (*Business Object*, *Enum Object*, *Business Exception*), schema, and file name.



The business type is also identified by an image:

- Manually created business objects
- Form business objects
- Imported business objects
- Auto-generated business objects
- Enum objects
- Business exceptions
- Click the business type name to edit it in the [Business Types Editor](#). Click the file name to view the file details.
- Select **Include Auto-Generated** to view the auto-generated business types.
The number of auto-generated business objects that exists in the application display in parenthesis.
 [Include Auto-generated \(11\)](#)
- Delete manually created business types.

Note:

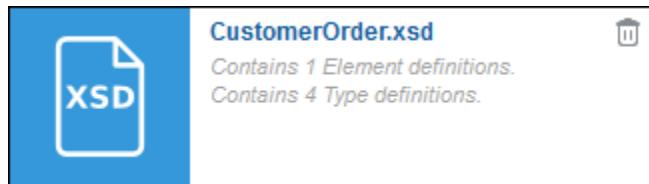
The **Delete** icon only appears for manually created business types. You can't delete business types that are being referenced by other artifacts.

Schema Files View

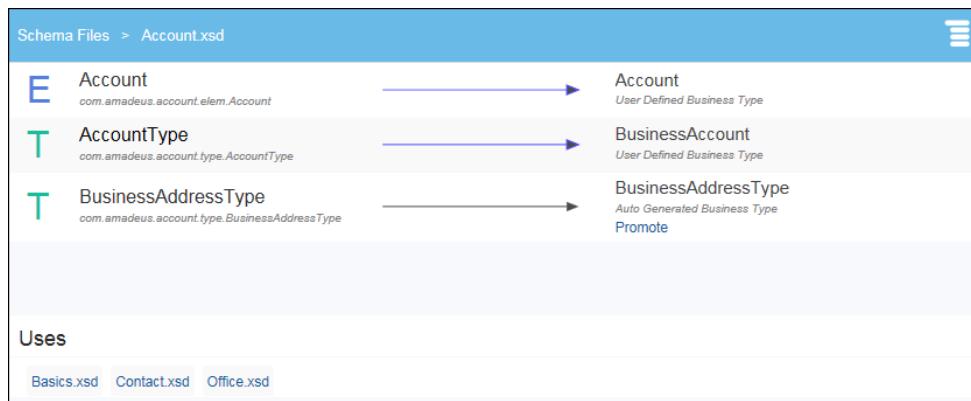
Use Schema Files view to perform the following tasks:

- [Import schema files.](#)
- Click **Views** to view the schema files in Grid or List format. Grid view is the default.

Each schema file displays its name, the number of elements and types it contains, and an XSD image.



- Click the schema file name to view its details, and perform the following tasks:
 - [Promote auto-generated business types to business objects.](#)
This is allowed as auto-generated business types aren't explicitly created by the user.
 - [Upload a new version of the schema file.](#)
 - View the details of other schema files that are imported from the current one.
(You can navigate from one to the other)



- Delete schema files.

Note:

You can't delete schema files that are being used by other business objects.

Working with Business Objects

Business objects let you group related types of data together to define the data structure required for your Process applications. Create business objects manually or base them on a XML Schema Definition (XSD). After defining business objects, you can use them to define data objects to store the data within your application.

Note:

Business objects are also created automatically when you design a form that isn't based on pre-existing data (form-first design).

See [Typical Workflow Using Form-First Design](#).

When manually creating business objects, you can define the hierarchical relationship between modules and objects, and the attributes used in the business object.

Note that you can't modify XSD-based business object properties. However, if you manually create a business object with an attribute pointing to another XSD-based business object, then you can modify the property values of that attribute.

Note:

If you create a business object based on a XSD, any changes made to the XSD (modify, add, delete elements) aren't reflected in the business object. After importing the modified XSD, you must delete the existing business object and then create a new business object on the modified XSD.

You can't create business objects based on in-line types defined in a WSDL file. However, read-only business objects are automatically created from the WSDL file when you create a web service connection. You can create data objects based on these read-only business objects.

Want to learn more about WSDL files and web service connections? See [Creating a Web Service Connection](#).

- Modules

Modules are containers that enable you to create a hierarchical structure. Each business object must be contained in a module. When you create a new business object, it's automatically created within the `BusinessData` module. You can create additional business objects within a module or create additional modules.

- Business objects

Within a module you can define one or more business objects. A business object can contain other business objects.

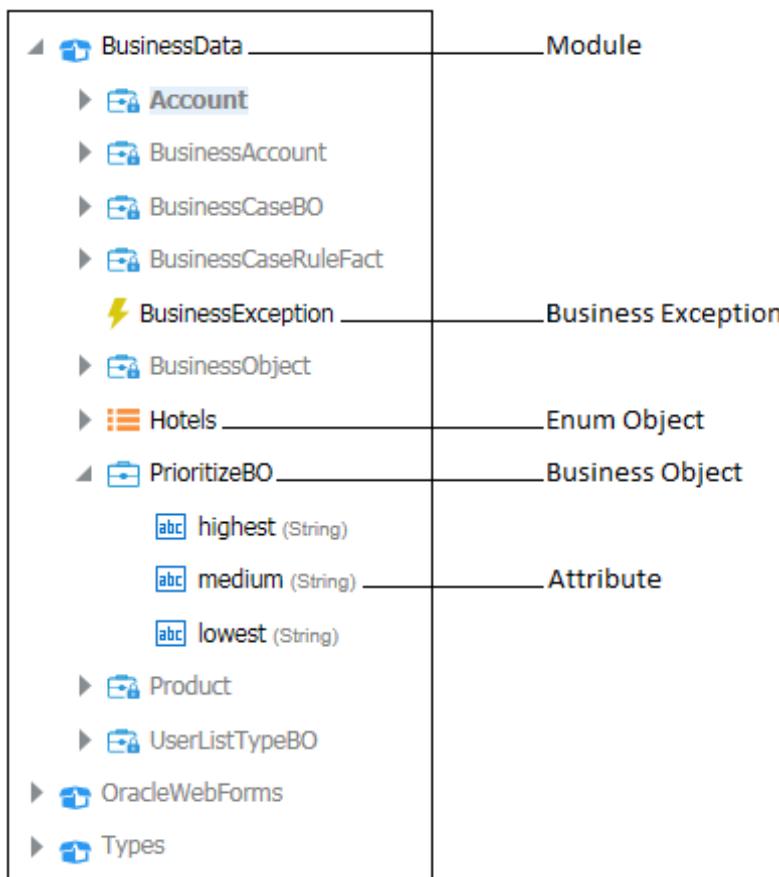
- Attributes

Attributes define particular pieces of process data that are stored and can be shared among process activities. Attributes represent a characteristic of a real-

world concept. For example, a business object that represents a person typically has `firstname` and `lastname` attributes.

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 application.

The following image shows an example of the hierarchical structure for the modules, business objects, and the business object attributes defined for an application.



An Enum object (enumeration) is a special type of business object that enables a data object to contain a set of predefined constants, such as one that includes the names of the days of the week (Sunday, Monday, Tuesday, and so on). See [Creating an Enum Object](#).

Business exceptions convey unexpected situations that can occur while running a business process. You can use Process to define business exceptions for your application. See [Creating a Business Exception](#).

Creating a Business Object

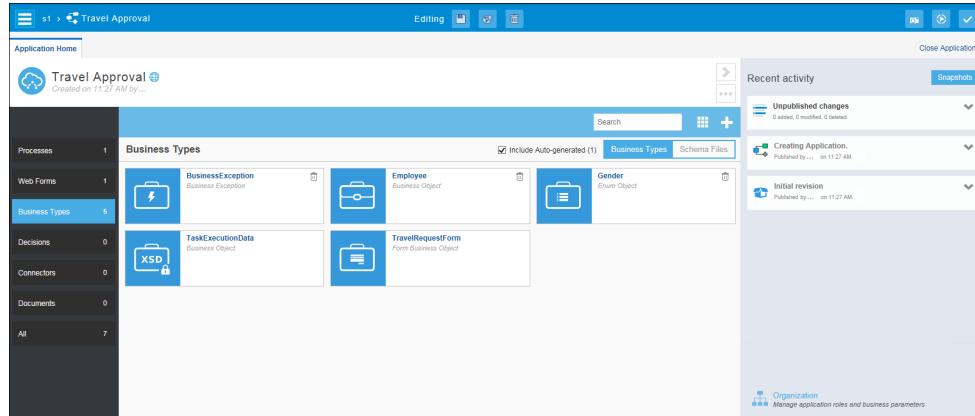
Create new business objects (complex data types) manually. When creating your business objects you define the hierarchical relationship between modules and business objects, and the attributes used in the business object.

Note:

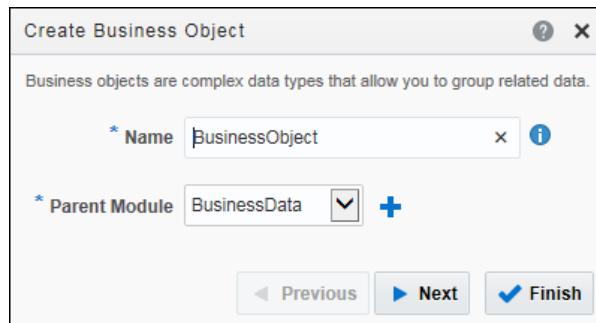
You can also import business objects based on XML schema (XSD) files. See [Importing a Business Object from XML](#).

To create a business object:

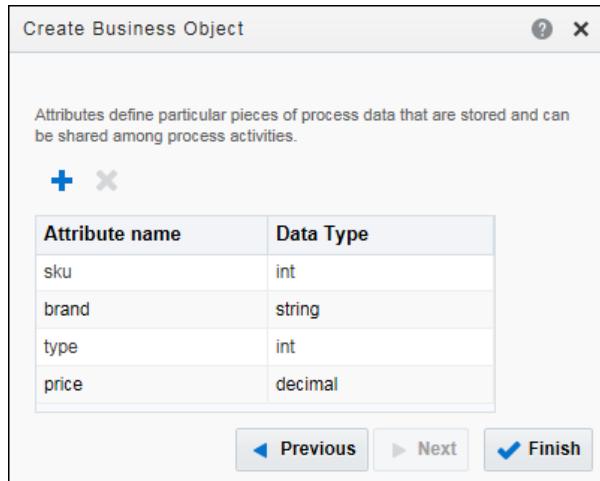
1. On the Application Home tab, click **Business Types**.



2. Click **New Business Type** , then select **New Business Object** to manually create a business object.
3. In the Create Business Object dialog box, enter a name for the business object or use the default name, select or add a parent module, and then:
 - Click **Finish** to create your new business object.
 - Click **Next** to add attributes.



4. Click **Add Attribute**  to add attributes. Enter a name, click **Change Type** to change the data type from the default value *String*, click **Add**. Click **Finish** after you add all your attributes.



Importing a Business Object from XML

You can import business objects (complex data types) that are based on an XML Schema (XSD) file. During the process you define the hierarchical relationship between modules and business objects.

To import a business object:

1. On the Application Home tab, click **Business Types**.
2. Click **New Business Type** , select **Import Business Object**, and then select **From XML Schema**.
3. In the Import Business Object from XML Schema dialog box, enter a name for the business object (or use the default), select or add a parent module, and select an element or type defined in a schema file, or click **Import Schemas from File**  to import an XSD or ZIP file. You can then highlight the required element or type from the list of elements and types.

 **Note:**

Primitive types as top level elements is not supported.

4. Click **Import** to create a business object based on an XSD file.

Importing a Business Object from JSON

You can import business objects (complex data types) that are based on a JSON instance or schema. During the process you define the hierarchical relationship between modules and business objects.

To import a business object from JSON:

1. On the Application Home tab, click **Business Types**.
2. On the Business Types page, click **New Business Type** , select **Import Business Object**, and then select **From JSON**.

3. In the Import Business Object from JSON Schema dialog box, enter a name for the business object, or click **Import from File**  to import a JSON file and the definitions contained within it.
4. Click **Import** to create a business object based on a JSON instance or schema.

The following are the limitations while creating business objects using JSON schemas:

- Imported JSON Schemas can contain references only to the definitions contained within it. External references aren't allowed.
- Only one file or JSON schema is accepted at a time.
- The use of *required* in the JSON schema doesn't impact the *Not Null* property of the resulting business object.
- Imported JSON cannot have fields that are not NCName compliant, for example, they cannot contain special characters like @ \$ % & / + , ; , whitespace characters, or different parenthesis. Furthermore a JSON field cannot begin with a number, a dot, or a minus character although they can appear later in the field.
- Process doesn't currently support JSON natively. The JSON schema is converted to an XML Schema before generating the business object.

 **Note:**

Because JSON isn't supported natively, the conversion to XML Schema must be valid. The field names (which are translated to XML Schema element names) must be compliant with XML Schema. For example, you can't use names like `some:name` because the colon conflicts with the namespace prefix declaration.

When converting a JSON schema to an XML schema, the following conventions are used:

- All valid JSON schema blocks must define either a `type`, `enum`, or `$ref` keywords. JSON schema blocks not identified as `type`, `enum` or `$ref` are ignored.
- The root JSON schema block can only be object type, or an `enum` block.
- Only root definitions are taken into account.
- Local definitions must be referenced by `#/definitions/<local_name>`.
- Keywords `allOf`, `anyOf`, `oneOf` in the JSON schema map to `xs:anyType` in the resulting XML schema.
- The `not` keyword isn't supported.

The following is an example of a JSON schema:

```
{  
  "id": "http://some.site.somewhere/entry-schema#",  
  "$schema": "http://json-schema.org/draft-04/schema#",  
  "description": "schema for an fstab entry",  
  "type": "object",  
  "required": [ "storage" ],  
  "properties": {  
    "storage": {
```

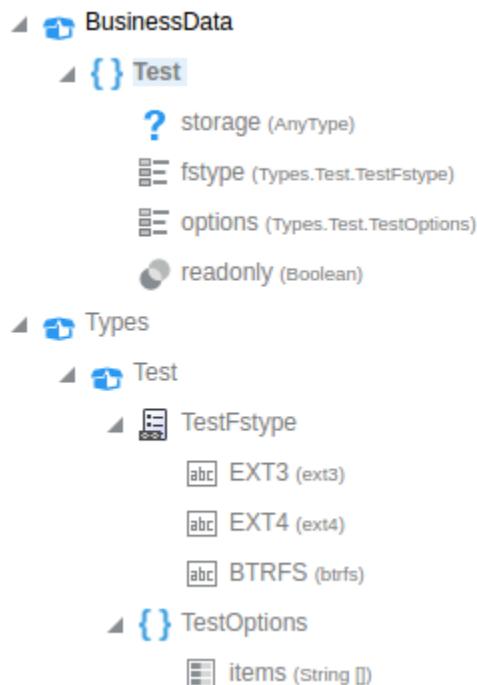
```

        "type": "object",
        "oneOf": [
            { "$ref": "#/definitions/diskDevice" },
            { "$ref": "#/definitions/diskUUID" },
            { "$ref": "#/definitions/nfs" },
            { "$ref": "#/definitions/tmpfs" }
        ],
        "fstype": {
            "enum": [ "ext3", "ext4", "btrfs" ]
        },
        "options": {
            "type": "array",
            "minItems": 1,
            "items": { "type": "string" },
            "uniqueItems": true
        },
        "readonly": { "type": "boolean" }
    },
    "definitions": {
        "diskDevice": {
            "properties": {
                "type": { "enum": [ "disk" ] },
                "device": {
                    "type": "string",
                    "pattern": "^\$dev/[^\$]+([^\$]+)+\$"
                }
            },
            "required": [ "type", "device" ],
            "additionalProperties": false
        },
        "diskUUID": {
            "properties": {
                "type": { "enum": [ "disk" ] },
                "label": {
                    "type": "string",
                    "pattern": "^\$[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}\$"
                }
            },
            "required": [ "type", "label" ],
            "additionalProperties": false
        },
        "nfs": {
            "properties": {
                "type": { "enum": [ "nfs" ] },
                "remotePath": {
                    "type": "string",
                    "pattern": "^\$([^\$]+)+\$"
                },
                "server": {
                    "type": "string",
                    "oneOf": [
                        { "format": "host-name" },
                        { "format": "ipv4" },
                        { "format": "ipv6" }
                    ]
                }
            },
            "required": [ "type", "server", "remotePath" ],
            "additionalProperties": false
        }
    }
}

```

```
        },
        "tmpfs": {
            "properties": {
                "type": { "enum": [ "tmpfs" ] },
                "sizeInMB": {
                    "type": "integer",
                    "minimum": 16,
                    "maximum": 512
                }
            },
            "required": [ "type", "sizeInMB" ],
            "additionalProperties": false
        }
    }
}
```

The resulting business object in Process appears as follows:



Importing XML Schema Files

Import and store XML Schema (XSD) files, and then use its complex types or elements to create business objects. You can also update the root XSD and all its dependencies.

Note:

Updating your schema files using this import feature is different from the update feature in the XSD details view where you can only update the current schema file. Want to learn more about how to update the current schema file? See [Uploading a New Version of an XML Schema File](#).

To import a XSD file:

1. On the Application Home tab, click **Business Types**.
2. On the Business Types page, click the **Schema Files** option, and then click  **Import Schema Files**.
3. In the Upload Schema File dialog box, select one of the following, and then click **Validate**.
 - **Upload from file:** Click **Browse** to browse for a XSD or ZIP file.
 - **Use URL:** Provide the URL for your XSD or Zip file.

A summary of the files to be added are shown if the validation is successful.

 **Note:**

If validation is unsuccessful, the validation errors or warnings are displayed. Validation warnings occur if some of the files that you want to upload override existing ones. In this case, you can upload and override existing files. Whenever an unexpected error occurs, the error is logged in the server log and the upload is cancelled. You can't import files that fail validation.

4. Optionally, click **Update** to update the root XSD and all its dependencies.
5. Click **Upload** to import your XSD file.

Uploading a New Version of an XML Schema File

Process lets you upload and validate new versions of existing XML schema (XSD) files.

To upload a new version of an XSD file:

1. On the Application Home tab, click **Business Types**.
2. Click **Schema Files**.
3. Click the name of the schema file to open it.
4. Click **Menu**  and select **Upload New Version**.
5. In the Upload New Version dialog box, select one of the following options:
 - **Upload from file:** Click **Browse** to browse for your XSD file.
 - **Use URL:** Provide the URL for your XSD file.
6. Click **Validate**.

A summary of the files to be added are shown if the validation is successful.

Note:

If validation is unsuccessful, then validation errors or warnings are displayed. Validation warnings occur if some of the files that you want to upload override existing ones. In this case, you can upload and override existing files. Whenever an unexpected error occurs, the error is logged in the server log and the upload is cancelled. You can't upload files that fail validation.

7. Click **Upload** to update your XSD file.

Promoting Schema Types and Elements to Business Objects

You can quickly create business objects (complex data types) that are based on specific XML Schema (XSD) types or elements by using the Promote feature available in the Schema Files — Details view.

To promote a schema type or element to a business object:

1. On the Application Home tab, click **Business Types**.
2. Click **Schema Files** to switch to the Schema Files view, which lists all schema files in the application.



3. Click the name of the required schema file to view its details.

Schema Files > Basics.xsd			
E	NotificationCriteria <small>com.amadeus.basics.elem.NotificationCriteria</small>	→	No Business Object Defined. Promote
E	NotificationCriteriaInstance <small>com.amadeus.basics.elem.NotificationCriteriaInstance</small>	→	No Business Object Defined. Promote
T	NotificationCriteriaInstanceType <small>com.amadeus.basics.type.NotificationCriteriaInstanceType</small>	→	No Business Object Defined. Promote
T	EmailType <small>com.amadeus.basics.type.EmailType</small>	→	EmailType Auto Generated Business Type Promote
T	NotificationCriteriaType <small>com.amadeus.basics.type.NotificationCriteriaType</small>	→	NotificationCriteriaType Auto Generated Business Type Promote
T	PhoneType <small>com.amadeus.basics.type.PhoneType</small>	→	PhoneType Auto Generated Business Type Promote
T	SystemIdentifierType <small>com.amadeus.basics.type.SystemIdentifierType</small>	→	SystemIdentifierType Auto Generated Business Type Promote
T	UserListType <small>com.amadeus.basics.type.UserListType</small>	→	UserListTypeBO User Defined Business Type
T	UserType <small>com.amadeus.basics.type.UserType</small>	→	UserType Auto Generated Business Type

This view shows the details of the schema file including:

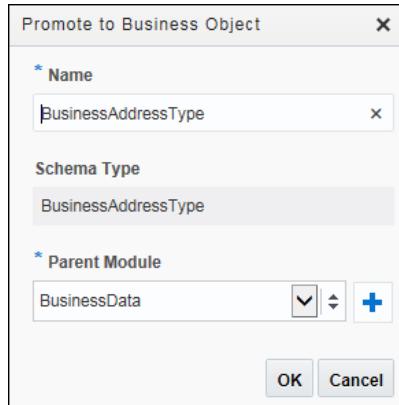
- User-defined business types

- Auto-generated business types
- Types and Elements with no business object defined

You can create business objects based on the types and elements that aren't already used for other business objects, as well as on the auto-generated business types because they're not explicitly created by the user.

4. Click **Promote** for the type or element you want to base your new business object.

The Promote to Business Object dialog box opens.



5. Enter a name for the new business object.

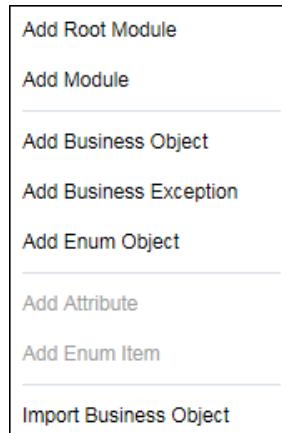
The field is populated with the name of the selected business type or element, such as *BusinessAddressType*. You can use this name or change it.

6. Select a parent module from the drop-down list or click **New** to create a new module and then click **OK** to create your business object.

Managing Business Types

After you create a business object, click its name to go to the Business Types — Data Structure page. From this page, you can add or change the type of a business object's attributes or add documentation. You can also add modules, business objects, business exceptions, Enum objects and Enum items, and import business objects to the data structure.

The business types and related items you can add are based on what you have highlighted in the hierarchy.



When you highlight a business type or related item in the hierarchy, its details appear in the pane on the right. You can edit the details or add documentation. For imported business objects, the type behind the business object and the file where that type is defined is also displayed.

The screenshot shows the Oracle Business Object hierarchy editor. On the left is a tree view of business objects under a root node 'BusinessData'. The 'BusinessAccount' node is selected and expanded, showing its children: Account, BusinessCaseBO, BusinessCaseRuleFact, BusinessException, BusinessObject, and Hotels. The 'Hotels' node is further expanded to show sub-items: RADISSON (1), HOLIDAYINN (2), COMFORTINN (3), and HOTEL6 (4). Other nodes like PrioritizeBO, Product, UserListTypeBO, OracleWebForms, and PrioritizeDecisionWF are also listed. On the right, a details pane for 'BusinessAccount' displays the following information:

- BusinessAccount**: The selected object.
- Description**: A placeholder for a description.
- Edit Documentation**: A button to edit documentation.
- Defined In**: Account.xsd: (Type) AccountType

A note at the bottom of the hierarchy view says: "Right-click a business object or related item in the hierarchy to delete it, move it, or collapse a section of the hierarchy."

Right-click a business type or related item in the hierarchy to delete it, move it, or collapse a section of the hierarchy.

Think before you delete. You can't recover a business object, module, or attribute that has been deleted. When you delete a business object or module that contains other related items, they're also deleted. However, if the deleted business object or module contains an attribute based on a business object, that business object isn't deleted.

Working with Data Objects

Data objects are, in general, the variables used to store the information used by your business processes. They're defined during the design and implementation stage of a process. In order to complete a process application, you must ensure that you have associated each activity with the data objects it requires.

At runtime, data objects contain information which may be altered as users interact with your business process. Running processes can store, access, and manipulate data. The values stored within a data object can also be used to determine the branch that a process takes.

Process supports data objects that are based on either simple or complex data types. Which type of data type you base your data object on depends on the type of data it must handle.

- **Based on Simple Data Types**

Data objects can be based on simple data types. These are the core data types common in most programming languages. The following table lists the basic data types supported by Process.

Type	Description
Bool	Represents the logical values <i>True</i> or <i>False</i> .
Int	Represents an integer. For example: 23, -10, 0.
Decimal	Represents a number that 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: <i>This is a string</i> .
Time	Represents a specific time expressed as: year-month-day hour:minute:second. For example: 2014-12-14 13:20:28.
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.

- **Based on Complex Data Types (Business Objects)**

Data objects based on complex data types (business objects) lets you create data structures that group together different types of data. Business objects allow you to create data structures from data objects based on simple data types.

For example, you can create a business object called employee that contains different types of data for employee such as id, name, and age. The relationship between data objects based on simple data types and data objects based business objects is similar to the relationship between classes and instances in the Java programming language. The following tables show this relationship.

Data Object	Simple Data Type
id	Int
name	String

Data Object	Simple Data Type
age	Int
sku	Int
price	Decimal
brand	String
type	Int

Data Object	Structure
Employee	<ul style="list-style-type: none"> — id (Int) — name (String) — age (Int)
Product	<ul style="list-style-type: none"> — sku (Int) — brand (String) — type (Int) — price (Decimal)

Before creating a complex data object, you must first define the business object that defines the data structure.

Creating a Data Object

You can create a data object based on a simple data object type or on a business object (complex data type). If you're creating a data object on a business object, you must create the business object first.

To create a data object:

1. On the Application Home tab, select a process, and then click **Data Objects**.
2. In the Data Objects dialog box, click **Add**.
3. Enter a name for your data object or use the default name, select a simple data object type or a data object based on a business object, click **Create**, then click **Close**.

 **Note:**

A default process data object with the name `TaskOutcomeDataObject` is automatically created for the human task outcome when a human task is created. The **Edit** and **Delete** links are disabled for this data object.

In the Data Objects dialog box, you can also edit or delete a data object. After editing or deleting a data object, validate your application to verify that there are no references to the changed or deleted data object.

See [Working with Data Objects](#).

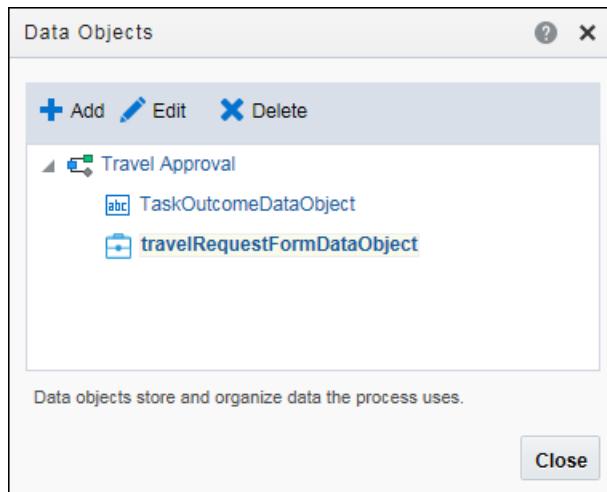
Editing or Deleting a Data Object

You can edit or delete data objects.

To edit or delete a data object:

1. Go to the Application Home tab and make sure you're in **Edit** mode if you're editing a shared application.
2. Open the business process where you want to edit or delete a data object.
3. Click **Data Objects** and expand the list of data objects for your business process.
4. Select the specific data object.

This step activates the **Edit** and **Delete** icons.



- To edit the data object, click **Edit**, then change the name or data type as necessary.
 - To delete the data object, click **Delete**.
5. Click **Close**.

After editing or deleting data objects, validate your application to verify that there are no references to the 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 type from an integer to a string, you must verify that all of the expressions that use the data object still function correctly. If they don't function correctly, the application doesn't validate.

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 don't remove references to the deleted data object, the application doesn't validate.

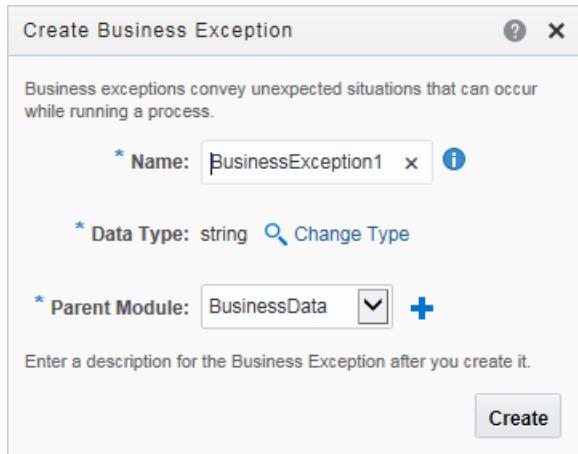
Creating a Business Exception

Business exceptions are used to convey unexpected situations that can occur while running a business process. Define business exceptions for your application and use them in an error end event that is triggered under a certain condition, such as an invalid Id or poor credit rating.

To create a business exception:

1. On the Application Home tab, click **Business Types** to open the Business Types page.

2. Click **New**, then select **New Business Exception** to open the Create Business Exception dialog.



3. Enter a name for the new business exception.

The field is populated with a default name, such as *BusinessException*, *BusinessException1*, and so on. You can use this name or change it.

4. Click **Change Type** if you want to change the type. The default value is *String*.
5. Select a parent module from the drop-down list, and then click **Create** to create your new business exception.

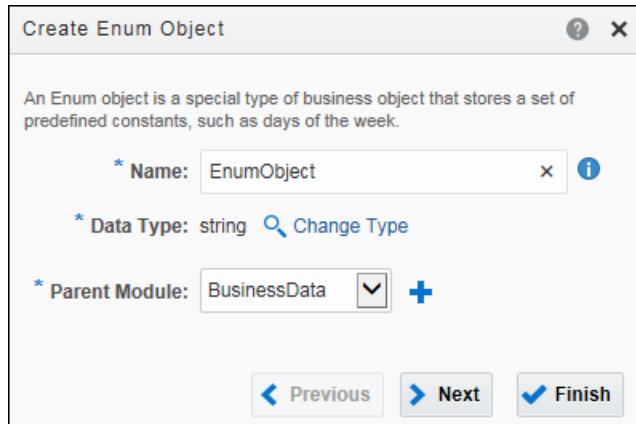
The parent module is *BusinessData* in most cases. If you're an advanced user, you can click **New** and create a new parent module.

Creating an Enum Object

You can manually create Enum objects for your application. An Enum object (enumeration) is a special type of business object that enables a data object to contain a set of predefined constants, such as one that includes the names of the days of the week (Sunday, Monday, Tuesday, and so on).

To create an Enum object:

1. On the Application Home tab, click **Business Types** to open the Business Types page.
2. Click **New**, then select **New Enum Object** to open the Create Enum Object wizard.



3. Enter a name for the new Enum object.

The field is populated with a default name, such as *EnumObject*, *EnumObject1*, and so on. You can use this name or change it.

4. Click **Change Type** if you want to change the data type of the Enum items. The default value is *String*.
5. Select a parent module from the drop-down list.

The parent module is *BusinessData* in most cases. If you're an advanced user, you can click **New** and create a new parent module.

You now have two options:

- Click **Finish** to create your new Enum object.
- Continue to the next step to add Enum items.

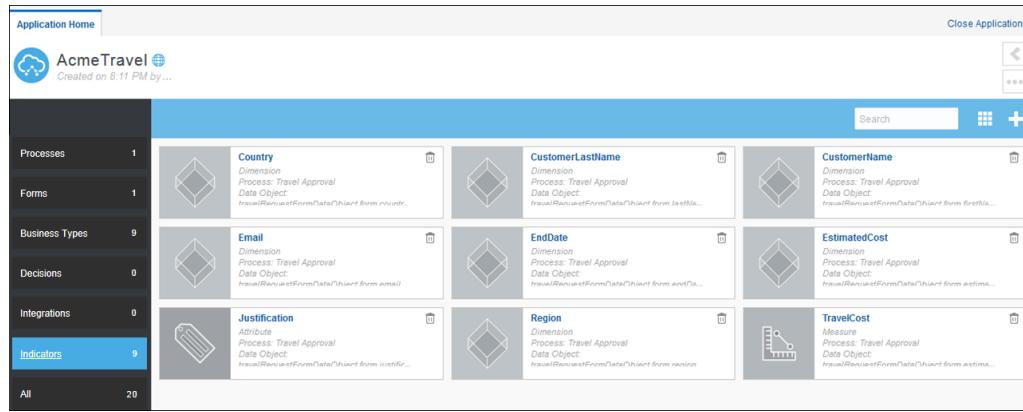
Perform the following steps to add Enum items.

6. Click **Next** to open the Enum Items dialog.
7. Click **New** to add an Enum item. Enter the name and value, then click **OK**.
Continue adding as many new Enum items as required.
8. Click **Finish** to create your new Enum object.

Working with Business Indicators

Business indicators provide a way to capture and display business metrics specific to your process. You designate selected data objects as business indicators, then plot them in business analytics dashboards. For example, for a sales quotation application, you might track key values that end users enter into a form, such as opportunity amount and region, then plot them in a chart displaying total opportunity amount by region.

The following image shows sample business indicators defined for an application:



You can create the following types of business indicators. Consider a data object's type when creating its business indicator.

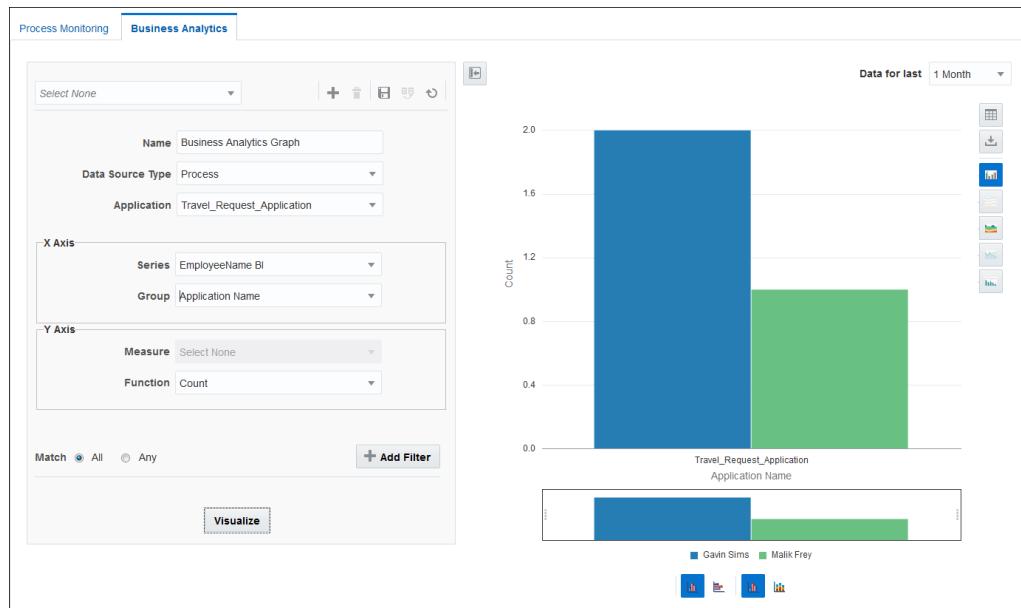
- Dimension indicators are available to plot as the X axis in charts, as a series and a group. Available dimension types include string, integer, decimal, Boolean, or date/time.
- Measure indicators are available to plot as the Y axis in charts, along with a function such as COUNT, SUM or AVG. They must be numeric (integer or decimal). You can set ranges for your numeric measure indicators.
- Attribute indicators are available to plot as filters in charts. Available attribute types include string, integer, decimal, Boolean, or date/time.

Note:

The X axis, the Y axis and the filters are available in the business analytics section in runtime.

After activating the application, you can select business indicators in combination to plot charts or graphs in runtime using business analytics options, as described in [Creating and Viewing Business Analytics Dashboards](#). You must be assigned the Process Owner role, the Administrator role, or the Analytics Viewer role to create and view business analytics in runtime. Business analytics options are available in production deployments.

The following image shows a chart plotted using the business indicators defined in the application.

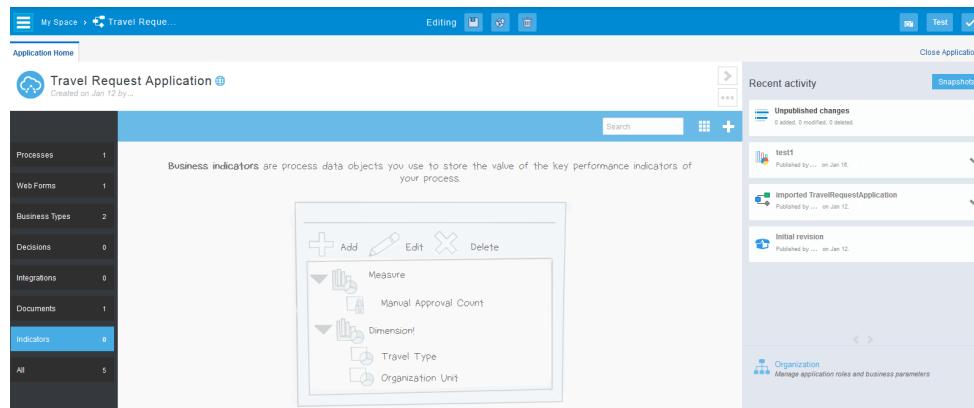


Creating Business Indicators

You can create business indicators to capture key metrics for use in custom visualizations, as described in Creating and Viewing Business Analytics Dashboard. You can create them using the Indicators pane or using data objects options in the process editor.

To create a business indicator from the Indicators pane:

1. Go to the Application Home tab and select **Indicators** on the left pane.



2. Click **New Business Indicator** and select a business indicator type to add.
 - **Attribute**: acts as a filter
 - **Dimension**: acts as the X axis
 - **Measure**: acts as the Y axis

 Note:

The X axis, the Y axis and the filters are available in the business analytics section in runtime.

3. Complete fields in the **Attribute**, **Dimension**, or **Measure** dialog box. In addition to entering a name for the indicator that will display in charts, identify the data object whose value will be captured.

Creating Business Indicators from the Process Editor

To add business indicators from the process editor:

1. Go to the Application Home tab, click **Processes** on the left pane, and then select a process.
2. On the process editor, click **Data Objects**.
3. In the Data Objects dialog box, expand the process and select a data object. Click **Indicator** and select a business indicator type to add.
4. Select the business indicator type, and enter a name (without spaces) in the **Name** field for the indicator, or leave the default name, and then click **OK**. The **Type**, **Process**, and **Data Object** fields are filled in by default. Click **Close** to close the data object dialog.

Deploying and Testing the Indicators

To deploy and test the business indicators in runtime:

1. [Deploy the application](#).
- You must deploy the application to a production environment to create business analytics reports. Business analytics options aren't available in test deployments.
2. Run multiple process instances defined in your application.
 3. Sign in as an administrator or process owner or the analytics viewer.
 4. Click **Dashboards** in the top menu, then select the **Business Analytics** tab.
 5. Create charts or graphs that use the business indicators, as described in [Creating and Viewing Business Analytics Dashboards](#).

Adding Dimension Business Indicators

Create dimension indicators to identify data objects whose values you want to display in business analytics dashboards.

Use dimension-type **business indicators** to capture values that can be grouped for display. Dimension indicators are plotted as the X axis in business analytics charts. You apply them as a series and a group in charts. You can also define ranges for numeric dimensions with continuous values.

1. In the Dimension dialog box, enter a name (without spaces) in the **Name** field for the indicator, or leave the default name. This name displays in charts and graphs, appended with BI.

The **Type** and the **Process** gets selected by default. Unless the application contains multiple processes, only one process is available.

2. In the **Data Object** field, browse and select the object whose data you want to capture in the indicator, and click **OK**. Note that only when you select the data objects, the type gets displayed.
3. Optionally, define range entries for a dimension indicator with a type of decimal or integer.

For example, for a dimension indicator that displays loan amounts, you might create multiple ranges such as the following:

Name	StartPoint	Range
High	50,000	>=50,000
Medium	25,000	25,000 ... 50,000
Low	0	0 ... 25,000

4. Deploy and test indicators, as described in [Creating Business Indicators](#).

Adding Measure Business Indicators

Create measure indicators to identify numeric data objects whose values you want to display in business analytics dashboards.

Use measure-type [business indicators](#) to capture numeric values that can be measured, such as amounts or counts in a loan approval process. Measure indicators are plotted as the Y axis in business analytics charts. You apply standard aggregation functions to them, such as COUNT, SUM, and AVG. They must be integers or decimals.

1. In the Measure dialog box, enter a name (without spaces) in the **Name** field for the indicator, or leave the default name. This name displays in charts and graphs, appended with BI.
The **Type** and the **Process** gets selected by default. Unless the application contains multiple processes, only one process is available.
2. In the **Data Object** field, browse and select the object whose data you want to capture in the indicator, and click **OK**. Note that only when you select the data objects, the type gets displayed.
3. Deploy and test indicators, as described in [Creating Business Indicators](#).

Adding Attribute Business Indicators

Create attribute indicators to identify data objects whose values you want to use as filters in business analytics dashboards.

Use attribute-type [business indicators](#) to capture values that aren't suited as measure or dimension indicators, such as transaction IDs. You can use them to filter information or refer to other information in the process.

1. In the Attribute dialog box, enter a name (without spaces) in the **Name** field for the indicator, or leave the default name. This name displays in charts and graphs, appended with BI.
The **Type** and the **Process** gets selected by default. Unless the application contains multiple processes, only one process is available.

2. In the **Data Object** field, browse and select the object whose data you want to capture in the indicator, and click **OK**. Note that only when you select the data objects, the type gets displayed.
3. Deploy and test indicators, as described in [Creating Business Indicators](#).

Associating and Manipulating Data

After you define business types and objects to store your application's data, you need to manage their use within flow elements.

Topics

- [Configuring Data Association](#)
- [Working with Transformations](#)
- [Working with Expressions](#)

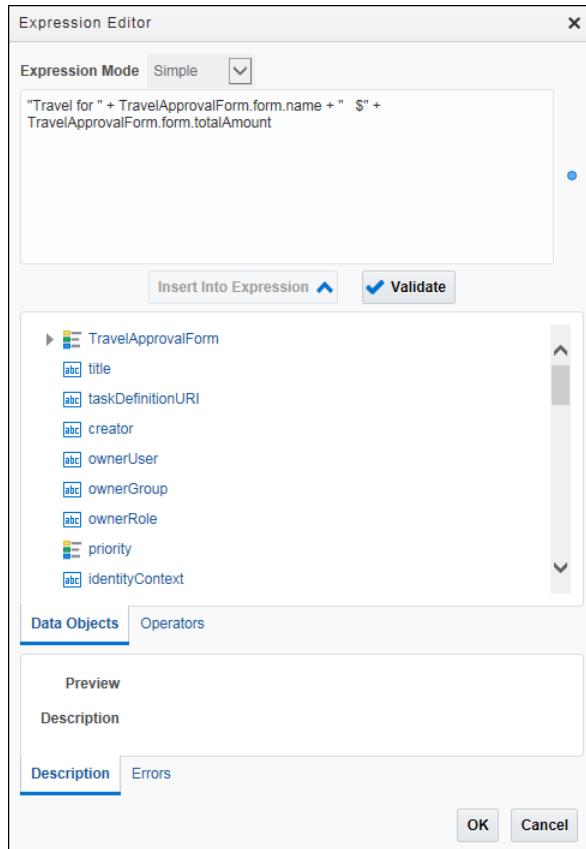
Working with Expressions

Use expressions to evaluate and perform calculations on data stored in data objects, using standard operators and functions.

Use expressions to change values passed to and from a sequence flow. You can create expressions when [configuring data associations](#) and [transformations](#), and when configuring properties for these flow elements:

- Human tasks: Use an expression to dynamically determine assignees
- Conditional sequence flows: Use an expression to define its condition
- Timer catch events: Use an expression to configure its time condition
- Notification tasks: Use expressions to define dynamic e-mail notification
- Sub-process tasks: Use an expression to conditionally call a subprocess

To create an expression, click the **Expression Editor**  icon or **Expressions Mode** field during [data association](#) or from the flow element's properties. The Expression Editor dialog box opens.



- In most cases, the **Expression Mode** field is set to Simple, but for some flow elements, Plain Text and XML Literal options are listed. The field below it displays the expression as you configure it.
- Create an expression by selecting the **Data Objects** or **Operators** tab, and selecting data objects and operators for your expression. Click **Insert Into Expression** to add a selected object or operator, or manually enter them.
- In the expressions field, enter a . (period) after a data object to view options available for its type (trigger a completion action).
- Click **Validate**, and results display in the expression field. The **Error** tab lists any errors found.

Using Functions

You can include the following functions in expressions, depending on data type. For example, you might use the string function to pass an integer value.

- **String** — text
- **int** — whole numbers
- **double** — decimal numbers
- **Boolean** — true or false
- **char** — single characters
- **byte** — eight bits, the smallest unit of data
- **short** — small whole numbers

- **long** — large whole numbers
- **float** — large decimal numbers
- **Date** — dates only
- **Time** — times only
- **DateTime** — dates and times

Function Guidelines and Examples

- If the field name is `inputDataObject`, and the type must be string, enter `string(inputDataObject)`.
- A field with a data type of `int` can contain integers only, or whole numbers. A field of type `double` or `float` can contain decimal numbers.
- In data association, if the input type is `int`, the output type can be any numeric data type, such as `int`, `double`, or `float`. However, if the output type is `int`, the input type must also be `int`, or an error occurs. Depending on needs, use the `round()`, `int()`, `floor()`, or `ceil()` functions.
- If the field name is `loanAppDataObject.form.income`, for basic forms, change it to `round(loanAppDataObject.form.income)` or `loanAppDataObject.form.income.round()` and for web forms, change it to `round(loanAppDataObject.income)` or `loanAppDataObject.income.round()`.

About Simple Expressions

Simple expressions are defined using a basic expression language and support. 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 business process. You can write expressions and conditions using the value of the data objects, but you can't explicitly modify the value within the data object.

Here are some examples of expressions using operators:

- `totalAmount - discount`
- `activationCount > 3`
- `unitsSold <= 1200`

Operator Precedence

Operator precedence defines the order in which the compiler evaluates operators. You can change operator precedence in an expression by using parentheses. In Process, the operator precedence is:

- Addition, Subtraction
- Multiplication, Division, Remainder
- Plus, Minus
- Less than, Greater than, Less than or equal to, Greater than or equal to
- Equals, Not equals
- Not
- Conditional And

- Conditional Or

Arithmetic Operators

Operator	Name	Description
+	Addition	Adds numeric data types; also concatenates strings
-	Subtraction	Subtracts numeric data types
*	Multiplication	Multiplies numeric data types
/	Division	Divides numeric data types
%	Remainder	Calculates the remainder of a division in which the divisor doesn't exactly divide the dividend
()	Precedence	Indicates the order of evaluation of an arithmetic expression

Unary Operators

Operator	Name	Description
+	Plus	Has no effect on the value of the numeric operand. Use it to explicitly indicate that a certain value is positive.
-	Minus	Negates an arithmetic expression.
!	Not	Logical complement operator. Negates the value of a Boolean expression.

Equality and Relational Operators

Operator	Name	Description
= or ==	Equal to	Returns true if the first operand is equal to the second operand
!=	Not equal to	Returns true if the first operand isn't 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

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

String Operators

Operator	Description	Usage Expression	Usage Result
+	String concatenation	"pine" + "apple"	"pineapple"

Operator	Description	Usage Expression	Usage Result
<code>==</code>	Equals	<code>"apples" == "apples"</code>	true
<code>!=</code>	Not equals	<code>"apples" != "oranges"</code>	true
<code>></code>	Greater than	<code>"word" > "work"</code>	false
<code>>=</code>	Greater than or equals	<code>"work" >= "work"</code>	true
<code><</code>	Less than	<code>"word" < "work"</code>	true
<code><=</code>	Less than or equals	<code>"work" <= "work"</code>	true
<code>contains</code>	Returns true if the first argument string contains the second argument string; otherwise returns false	<code>"caramel".contains("ram")</code>	true
<code>endsWith</code>	Returns true if the first argument string ends with the second argument string; otherwise returns false	<code>"immutable".endsWith("table")</code>	true
<code>length</code>	Returns the number of characters in a string	<code>"house".length()</code>	5
<code>lowerCase</code>	Returns a string with all the characters in the argument converted to lower-case representation	<code>"Example".lowerCase()</code>	"example"
<code>startsWith</code>	Returns true if the first argument string starts with the second argument string, otherwise returns false	<code>"caramel".startsWith("car")</code>	true
<code>substring</code>	Returns the substring of the first argument starting at the position specified in the second argument and continuing to the end of the string	<code>"care".substring(1)</code>	"are"
<code>substring</code>	Returns the substring of the first argument starting at the position specified in the second argument with length specified in the third argument	<code>"care".substring(0,3)</code>	"car"
<code>upperCase</code>	Returns a string with all the characters in the argument converted to upper-case representation	<code>"Example".toUpperCase()</code>	EXAMPLE

Integer Operators

Operator	Description	Usage Expression	Usage Result
<code>+</code>	Addition	<code>2 + 8</code>	10
<code>-</code>	Subtraction	<code>7 - 4</code>	3
<code>*</code>	Multiplication	<code>3 * 4</code>	12
<code>/</code>	Division	<code>3 / 2</code>	1.5

Operator	Description	Usage Expression	Usage Result
%	Remainder	3 % 2	1
==	Equals	12 == 13	false
!=	Not equals	12 != 13	true
>	Greater than	15 > 16	false
>=	Greater than or equals	15 >= 15	true
<	Less than	12 < 10	false
<=	Less than or equals	12 <= 12	true
abs	Returns the absolute value of a number	- 6	6

Non-integer Operators

Operator	Description	Usage Expression	Usage Result
floor	Returns the largest (closest to positive infinity) number that isn't greater than the argument and is an integer	floor(5.60)	5
ceil	Returns the smallest (closest to negative infinity) number that isn't less than the argument and is an integer	ceil(5.60)	6
round	Returns the number that is closest to the argument and is an integer	round(5.60)	6

Date and Time Operators

Operator	Description
+	Addition (valid only when the second argument is a duration)
-	Subtraction (valid only when the second argument is a duration)
==	Equals
!=	Not equals
>	Greater than
>=	Greater than or equals
<	Less than
<=	Less than or equals
format	Returns the formatted string of date-time using the provided format picture
year	Returns a number representing the year component of the date-time argument
month	Returns a number representing the month component of the date-time argument
day	Returns a number representing the day component of the date-time argument
hours	Returns a number between 0 and 23, both inclusive, representing the hours component of the date-time argument
minutes	Returns a number between 0 and 59, both inclusive, representing the minutes component of the date-time argument
seconds	Returns a number between 0 and 59, both inclusive, representing the seconds component of the date-time argument

Operator	Description
timezone	Returns an interval value, representing the time offset from UTC

Boolean Operators

Operator	Description	Usage Expression	Usage Result
==	Equals	true == true	true
!=	Not equals	true != false	true
and	Conditional — And	true and false	false
or	Conditional — Or	true or false	true
not	Logical complement operator, inverts the value of a Boolean expression.	not true	false

Duration Operators

Operator	Description
==	Equals
!=	Not equals
>	Greater than
>=	Greater than or equals
<	Less than
<=	Less than or equals

Base64Binary Operators

Operator	Description
==	Equals
!=	Not equals

Array Operators

Operator	Description
[]	Access a particular element into the array
==	Equals
!=	Not equals
length	Returns the number of elements contained within the array

Other Operators

Operator	Description
==	Equals
!=	Not equals

Special Constants

Constants	Description
null	Null value
true	Logical true
false	Logical false
'now'	Current dateTime

Casting

In some cases, it could be desirable to bypass the type validation in order to assign types that aren't necessarily compatible. For example, you may want to assign an 'int' value to a 'string' one and in order to do that you can use the conversion operation like this:

`<conversionTypeName> (<valueToConvert>)`

where the 'conversionTypeName' is the type you want to see as the value.

Here are some conversion examples:

- `string(myIntDO)`
- `int(myStringDO)`
- `duration(mystringDO)`

Note:

You can only cast to primitive types, so the 'conversionTypeName' will only accept those that have a valid value.

Assigning two values that are incompatible will result in a runtime error.

Identity Service

Function	Description	Available	Usage Function Prototype	Usage Example
getManager	Returns a string containing the manager of the specific user	Process wide	IdentityService.getManager(<userName:string>): string	IdentityService.getManager("wfaulkner")
getManager	Returns a string containing the manager of a certain user in the realm specified	Process wide	IdentityService.getManager(<userName:string>,<realm:string>): string	IdentityService.getManager("wfaulkner", "myRealm")

Human Task

Function	Description	Available	Usage Function Prototype	Usage Example
getPerformer		Assignee Selection	HumanTask.getPerformer(): string	HumanTask.getPerformer()
getLastPerformer		Assignee Selection	HumanTask.getLastPerformer(): string	HumanTask.getLastPerformer()

Form

Function	Description	Available	Usage Function Prototype	Usage Example
getWebform	Returns a base64 encoded value representing the PDF image of the specified webform data object	Process wide	Form.getWebform(<webFormDataObject:catalogObject >): base64	Form.getWebform(myWebformDO)
getWebform	Returns a base64 encoded value representing the image of the specified webform data object in the format passed as the second argument (valid values are "PDF" or "PNG")	Process wide	Form.getWebform(<webFormDataObject:catalogObject >,<format:string>): base64	Form.getWebform(myWebformDO, "PNG")
getWebform	Returns a base64 encoded value representing the image of the specified webform data object having the name determined in the third argument in the format passed as the second argument (valid values are "PDF" or "PNG")	Process wide	Form.getWebform(<webFormDataObject:catalogObject >,<format:string>,<webFormName:string>): base64	Form.getWebform(myWebformDO, "PNG","myWebformDO")

Social Service

Function	Description	Available	Usage Function Prototype	Usage Example
getConversationProperty	Get some property for a specific conversation, which can be one of the following: Id	Process wide	SocialService.getConversationProperty(<propertyName: string>,<conversationName: string>): string	SocialService.getConversationProperty("Id","myConversation")

Document Service

Function	Description	Available	Usage Function Prototype	Usage Example
getDocumentAssetProperty	Get some property for a specific document asset, which can be one of the following: Id, Type	Process wide	DocumentService.getDocumentAssetProperty(<property Name:string>,<documentAssetId:string>,<documentAssetName:string>): string	DocumentService.getDocumentAssetProperty("Id","myFolder")

Configuring Data Association

Data association refers to the flow of data within a process. Use the Data Association editor to define input and output for flow elements that need them.

A data association involves a source and a target, where the source provides a value or an expression to be assigned to the target. An approval human task, for example, needs both input and output data association.

- On the input side, it needs data input into the activity (referred to as its payload).
- On the output side, after the activity has just finished, it needs output from the activity to data objects, to store results for use elsewhere in the process.

To configure data association:

1. In the process editor, select a flow element that needs data association, such as a human task or service task, and click the **Data Association** button in the upper right. (This button is active when the selected flow element needs association. See [Data Association Tips](#).)

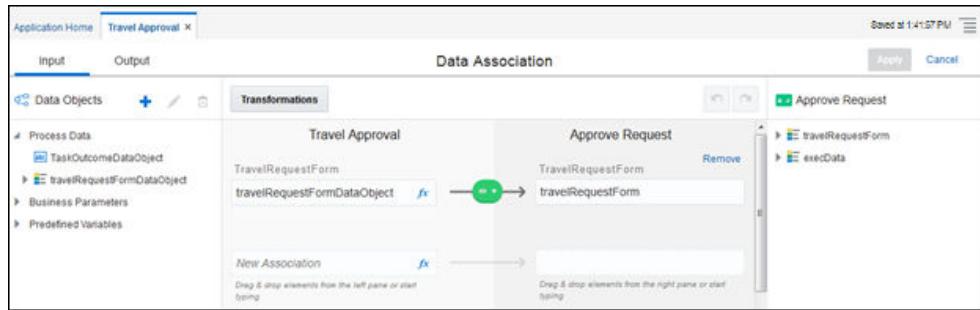
The Data Association editor opens. Use its center pane as a canvas for mapping associations, by dragging and dropping elements from the left and right panes.

Any data associations already configured display below **Data Association**. You use the empty pair of fields to create a new association.

Note:

Process creates data associations for you in some cases, such as when you create a human task or start event with a form.

- a. Notice that the **Input** tab is selected. The left pane displays source objects (**Data Objects**) in an expandable tree. The right pane (indicated by the task's icon) displays the payload, or entry parameters the activity needs to perform its function.
- b. Click the **Output** tab and notice how the left and right panes change. Now the left pane displays the task's payload, and the right pane displays the variables available in the process.



2. Click the **Input** tab again, and begin creating a new association by dragging an input object from the Data Objects tree and dropping it in the input field titled *New Association*. You can also begin typing and select from autocomplete options that appear. To specify attributes within objects, enter a . (period) and select from the list that displays.

The input source provides the value or expression to assign to the target. You select variables related to the process (such as data objects or business parameters) to map to the specific parameters the activity needs to perform its function (its entry parameters).

3. If needed, click the **fx** (Expression) icon and use the Expression Editor to create an input expression using standard functions and operators. See [Working with Expressions](#).
4. Complete the association by dragging an output object from the payload tree and dropping it in the output field.

Process validates the new association and displays a green association icon if valid, or a red association icon and error if invalid:



Valid data association



Invalid data association

See [Data Association Tips](#) for additional information about configuring associations.

5. If the association is invalid, optionally create a transformation to convert its associated attributes.

A transformation maps mismatching data types. See [Working with Transformations](#).

6. If needed, create additional associations and reorder them.

Drag and drop an association to move it up or down in the associations list. Associations are executed in the order in which you position them. For example, if multiple associations assign a value to the same object, the last assigned value is used.

As you edit, click the **Undo** or **Redo** buttons as needed.

7. Click the **Output** tab, and create output data associations. You can think of the output as though the activity has just finished, and likely contains results that you'd like to store for use elsewhere in the process.
8. Click **Apply** to save the data associations.

You can save invalid associations and fix them later. Note that you can't play or deploy applications that contain validation errors.

Data Association Tips

Use the Data Association Editor to add, edit, delete, and reorder data associations.

Below are tips for [configuring data associations](#).

- The following flow elements need data association. Note that some flow elements have only output, such as a start form event, which captures the data end users enter into the form.
 - Human tasks
 - System tasks, except abstract and notification tasks
 - Events, except timer catch, error boundary, and terminate end events
- You can access data association in any of these ways from the process editor:
 - By clicking the **Data Association** button in the upper right.
 - By clicking a flow element on the process, clicking the menu icon, and choosing **Open Data Association**.
 - By opening the Properties pane and clicking the input (right) or output (left) arrow in the lower left of the pane.
- In the Data Association Editor, click the **Input** tab to define data input into the activity, and the **Output** tab to define data output resulting from the activity.
- Under **Data Objects**, you can create, edit, and delete data objects if needed without exiting the data association editor.
- Dropping an object into an association field that already contains a value replaces its value. If you drop an object into an input field that ends with an operator, the object name is appended to the field's contents.

Data Association for Enums

The Data Association Editor provides flexibility in associating enums (enumerations) and primitive types.

 **Note:**

You can also transform enum items. See [Creating Transformations Between Enum Items](#).

Follow these guidelines for associating [enum objects](#).

- **You can assign an enum type to a primitive type.**

In this case, you're assigning a source that includes a limited set of values to a target that includes a full set of values. Note that the types must be compatible; for example, you can't assign an integer enum data object to a string data object.



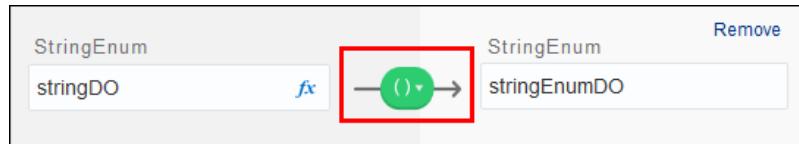
- **Use caution when assigning a primitive type to an enum type.**

In this case, you're assigning a source that includes a full set of values to a target that includes a limited set of values. The data association editor allows the assignment, but displays an **Automatic Casting** icon as a caution that runtime issues may result.



Automatic Casting association icon

For example, an error might be triggered when data from an automatically cast association is passed to an external service. This same behavior occurs when using explicit casting, as described in [Working with Expressions](#).

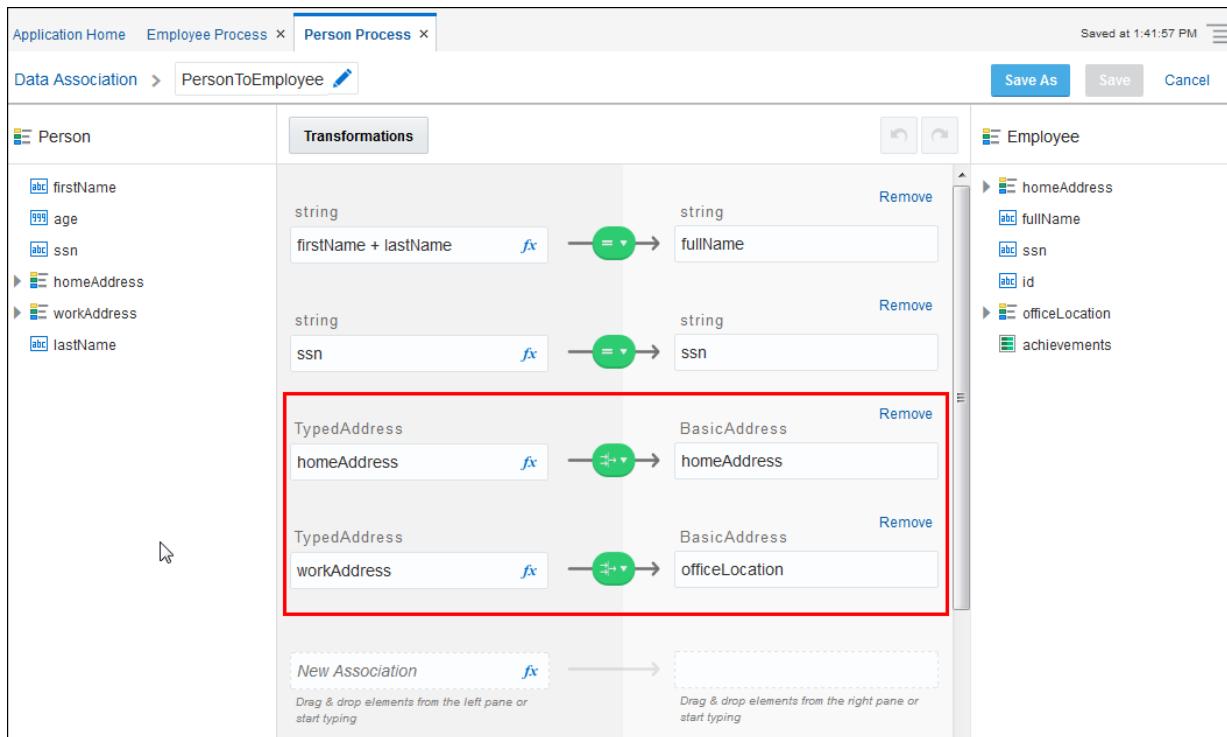


Working with Transformations

A transformation is a special type of data association between input and output data types that don't match. A transformation both associates input and output objects and maps their types. And because transformations are reusable, you can configure them once for use throughout the application, then apply them as needed.

Transformations are particularly useful for matching slightly differing data attributes, such as address or employee data. For example, your process might connect to a service that returns employee data with slightly different attributes that need to be mapped. In the illustration below, the bottom two associations are transformations between mismatched address objects.

You can use transformations with arrays, and they're applied to each item in the array.



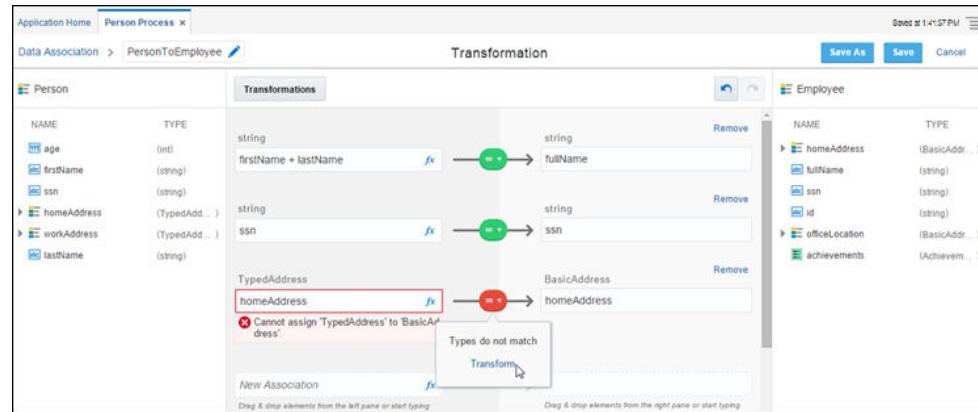
Note that transformations have their own icon, which differs slightly from the data association icon. As with data associations, the icon is green when valid. When invalid, the icon is red and its error displays.

You [create and apply transformations](#) while [configuring data associations](#). You can either transform a selected data association or create a transformation and then apply it. You can even create transformations within transformations.

Creating, Applying, and Editing Transformations

You create and apply transformations while creating data associations.

1. In the process editor, follow the steps to [configure a data association](#). Select a flow element that needs data association, click the **Data Association** button, and map objects to the association input and output fields.
2. Click the data association icon, and select **Transform**. (The association can be valid or invalid.)

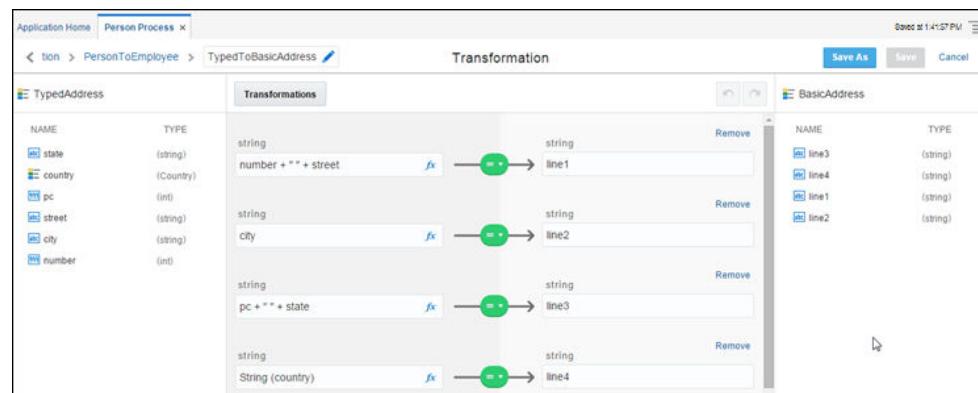


- In the Transformations window (named based on the objects you selected), click **Create**. In the New Transformation dialog box, enter a name, and click **Create**.

The Transformation Editor displays. It looks and functions much like the Data Association Editor, except that its left and right panes reflect the selected objects to transform.

- Configure a transformation between the objects by dragging and dropping elements to the input and output fields. You can also begin typing and select from autocomplete entries that display.

If needed, click the **fx** (Expression) icon and use the Expression Editor to create an expression for an input field. See [Working with Expressions](#).



- Click **Save** to save the transformation. You can use the **Save As** button to create a new transformation based on the selected one.

Process applies and validates the new transformation, displaying a green transformation icon if valid, or a red transformation icon and error if invalid. Note that you can save an invalid transformation and fix it later.

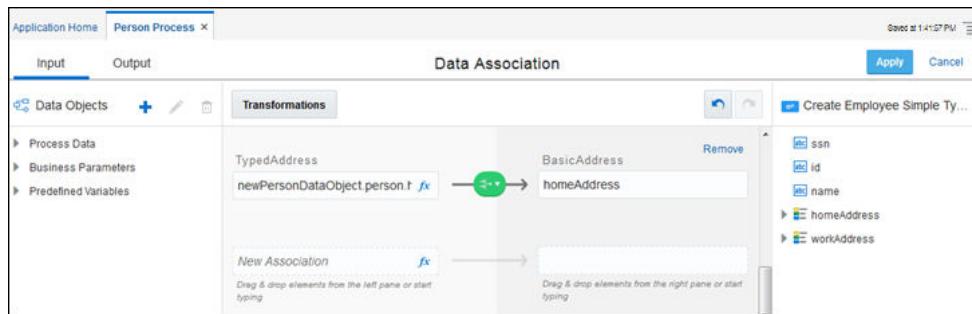


Valid transformation



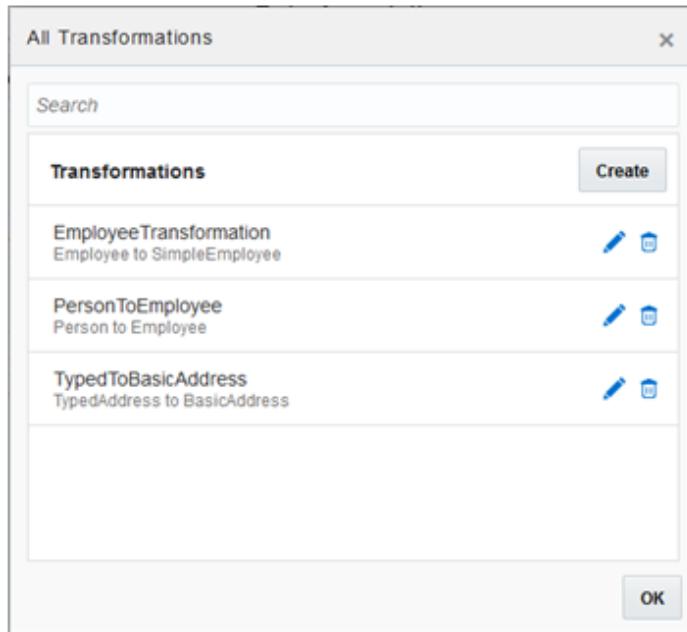
Invalid transformation

- Click the **Data Association** link to close the transformation editor and return to the data association editor.



7. Create, edit, or change transformations as needed.

At any time, you can display a list of all transformations defined for the application by clicking the **Transformations** button in the Data Association Editor. This displays the All Transformations window, which includes options for searching for, creating, editing, and deleting transformations. If creating a transformation from this window, you must select the input and output objects to transform from the application data objects listed in **From** and **To** fields, then apply the transformation to a data association.



- To apply a transformation to a data association, click its icon in the Data Association Editor and select the **Transform** link. From the list of transformations available for the selected data types, select a transformation to apply and click **OK**.
- To apply a different transformation, click a transformation icon in the Data Association Editor and select the **Change** link. From the list of transformations available for the selected data types, select another transformation and click **OK**.
- To edit a transformation, click its transformation icon in the Data Association Editor, select the **Edit** link, and make edits in the Transformations Editor.
- To delete a transformation, click the **Transformations** button, select it from the Transformations list, and click **Delete**. If the transformation is in use, a

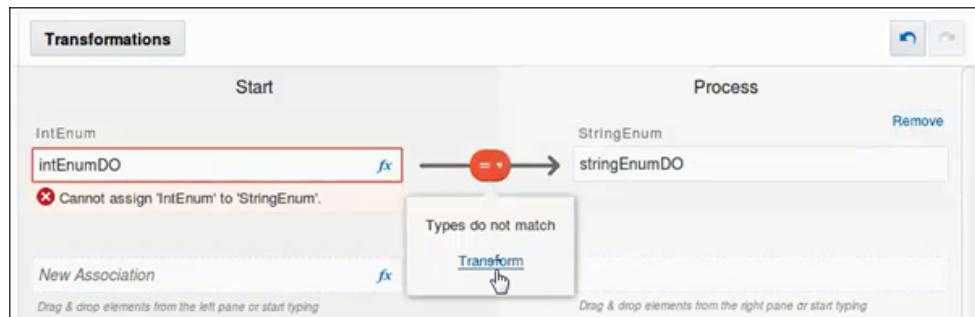
warning lists where it's in use and will be removed if you proceed with the deletion.

Creating Transformations Between Enum Items

You can create and apply transformations between enum items when creating data associations.

Enum transformations work similarly to [regular transformations](#), with a few differences highlighted below. Note that you can create enum transformations between enums only.

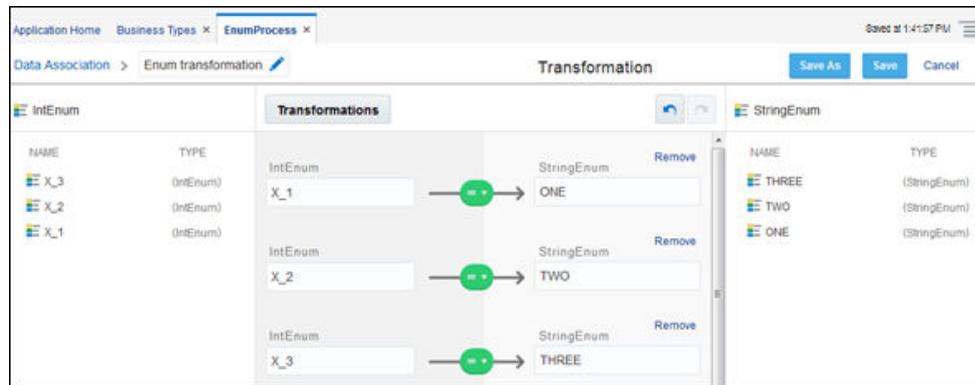
1. In the process editor, follow the steps to [configure a data association](#): select a flow element that needs data association, and click **Data Association**.
2. In the data association editor, drag and drop source and target enums. Click the invalid association icon, and select **Transform**.



3. In the Transformations window (named based on the objects you selected), click **Create**. In the New Transformation dialog box, enter a name, and click **Create**.

The Transformation Editor displays. It looks and functions much like the Transformation Editor does when enum types aren't selected, with several differences:

- Instead of Input and Output tabs on each side of the transformation, you see the enums and their enum items to map individually.
- No expression editor icon is shown, because simple expressions aren't allowed in enum transformations.



4. Configure the transformation by dragging and dropping enum items to map source and target fields. For example, you might convert X_1 from IntEnum to ONE from StringEnum. Note that you can't create transformations between enum items.

5. Click **Save** to save the transformation. You can also click **Save As** to create a new transformation. If needed, click **Apply** to apply the transformation you created to the data association.

Process validates the new transformation, displaying a green transformation icon if valid, or a red transformation icon and error if invalid.

10

Creating Decisions

In Process, create decision models to automate the decision logic inherent in your business processes. As part of creating a decision model, add and order decisions, define their input, model their logic using simple or complex types, test them, and expose them as services for use in Process applications.

Note:

Process provides two editors related to decisions and rules: the **Decision Model editor** (recommended, described below), and the **Oracle Business Rules editor** described in [Creating Business Rules](#).

Topics

- [How do decisions work in process applications?](#)
- [Ready to create a decision model?](#)
- [Working with Decision Models](#)
- [Creating a Decision Model](#)
- [Adding and Ordering Decisions](#)
- [Understanding FEEL \(Friendly Enough Expression Language\)](#)
- [Defining Decision Input and Type](#)
- [Modeling Decision Logic](#)
- [Testing Decisions](#)
- [Exposing Decisions as Services](#)
- [Deploying and Managing Decision Model Snapshots](#)
- [Adding Decisions to Applications and Processes](#)
- [Importing and Exporting Decision Models](#)

How do decisions work in process applications?

Use the Decision Model editor as a canvas to define, test, and output decision logic for process applications. The editor supports the Decision Modeling and Notation (DMN) standard, version 1.1, and uses FEEL (Friendly Enough Expression Language) to make decision modeling easier and more intuitive.

Automate and reuse decisions throughout your business processes

In business, decisions are everywhere. Should this loan application or document change be approved? Should emergency vehicles be dispatched to this incident? How many bonus shopping points is this shopping cart worth?

Use the decision model framework to express a full range of automated decisions. Model your decisions as a tree of simple decisions, each automated using decision tables and simple expressions instead of production rules. Enter expressions in a simple standard expression language without worrying about quotation marks or special formatting. Then deploy and use your decision models in one or more applications, and easily modify them as needed.

If you're new to the editor, [start here to learn the basics](#) by creating and configuring a decision model and then test it before deploying it.

Want to learn more about creating decision tables or complex logic structures such as functions, relations, or contexts? See [Modeling Decision Logic](#). Wondering how decision models are deployed and used in applications? See [Working with Decision Models](#).

Ready to create a decision model?

Ever leave the house and wish you'd better prepared for the weather?

Let's create a process application to fix that. When prompted, you input the temperature and rain forecast, and the application decides whether you should bring:

- An umbrella, for warm and rainy weather
- A raincoat, for cold and rainy weather
- An overcoat, for cold and dry weather
- Nothing, for warm and dry weather

Using this example, let's explore the entire decision model development life cycle—from creating a decision model, defining its decision logic, activating it on the DMN server, and using a decision service in a process.

- [Create a Simple Application](#)
- [Create a Decision Model](#)
- [Add Decisions](#)
- [Define Input](#)
- [Model Decision Logic](#)
- [Test Your Decisions](#)
- [Create a Service](#)
- [Deploy a Decision Snapshot](#)
- [Use the Decision in Your Application](#)
- [Map Data to and from the Decision](#)
- [Try Your Decision in Runtime](#)

Create a Simple Application

First, let's create an application for the decision. Its process starts with a web form, which has two input fields where you enter temperature and rain forecast numbers. It then uses a decision element to call the decision service. Lastly, a human task

element displays the decision result, which tells you what to bring for the day based on the weather.

1. Create an application.

In the Composer Home page, click **Create** and select **New Application**. Enter a name (**What To Bring**) and click **Create**.

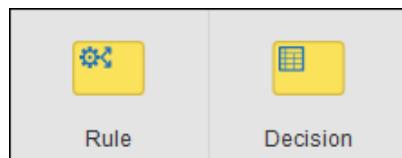
2. Create a process.

In the Create a Process pane, click **Start with a form**, enter a process name (**Decision Process**), and click **Create**.

3. Add a decision and human task.

On the Decision Process tab, expand **System** on the Elements palette, drag a **Decision** element to the process, and drop it after the start event.

Notice that the System category contains both Rule and Decision elements.

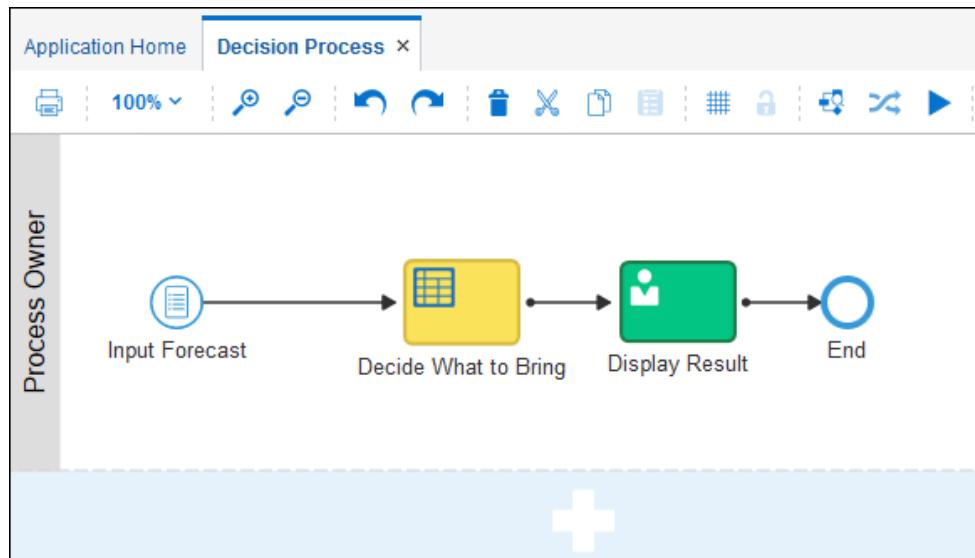


Rule elements correspond to rules created using Oracle Business Rules. See [Creating Business Rules](#).

Expand **Human** on the palette, drag a **Submit** element to the process, and drop it after the decision element.

4. Rename the flow elements.

Double-click the flow elements and rename the start event to **Input Forecast**, the decision to **Decide What to Bring**, and the human task to **Display Result**.



5. Add a web form.

Select the **Input Forecast** element, click **Menu** and select **Open Properties**. In the **Title** field, enter a name (**Forecast**) to display for the application.

Click **Create New Form** + next to the **Form** field and select **New Web Form**. Enter `InputForecastWebForm` in the **Name** field, select the **Open Immediately** check box, and click **Create**.

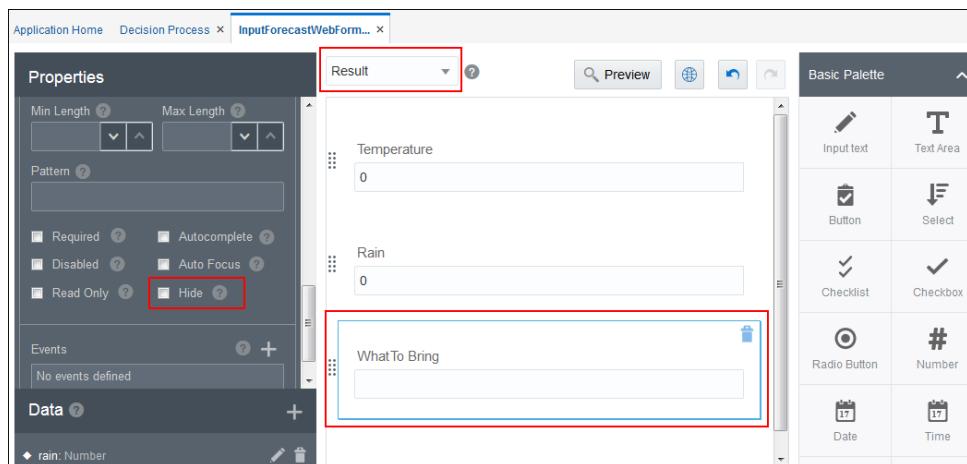
6. Add the forecast input fields to the web form.

From the Basic palette, drag and drop two **Number** controls and one **Input Text** control onto the canvas. Select each control and edit its **Label** field in the Properties pane, changing the number controls to **Temperature** and **Rain**, and the text control to **What To Bring**.

Select the **What To Bring** control, scroll down the General tab, and select the **Hide** check box. Notice that the control becomes dimmed to indicate it's hidden.

7. Add a presentation to display results.

Click the form outside a control. On the Form tab, click **Add Presentation** + next to **Presentations**. Enter a name for the new presentation (**Result**), select **Main** in the **Based on** field, and click **Create**. On the new Result presentation, select the **What To Bring** control and deselect the **Hide** check box.



8. Save the application.

9. Return to the process by clicking the **Decision Process** tab. Open the properties for the **Display Result** human task. Click **Browse** 🔎 next to the **Form** field and select **InputForecastWebForm**. In the **Presentation** field, select **Result**.

Create a Decision Model

A decision model does what its name implies: it lets you define how decisions will work. It contains a list of decisions, input data, and other definitions.

To create a decision model:

1. Return to the Composer Home page by clicking the Spaces link (**My Space**) in the toolbar.

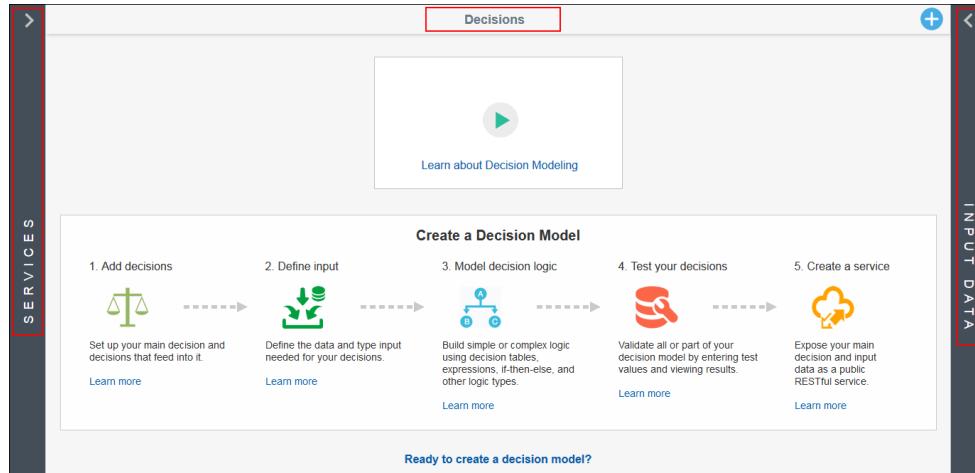
Spaces can contain applications and decision models. You can use a decision model in multiple applications.

2. Click **Create** and select **New Decision Model**.

You can use a decision model in multiple applications.

3. In the Create Decision Model dialog box, enter a name for the model (*What To Bring Decision Model*) and click **Create**.

The decision model editor opens in a new page named with the title you entered. Notice that the editor contains a central pane labeled Decisions on which you construct decisions, with expandable Services and Input Data side panes.



Add Decisions

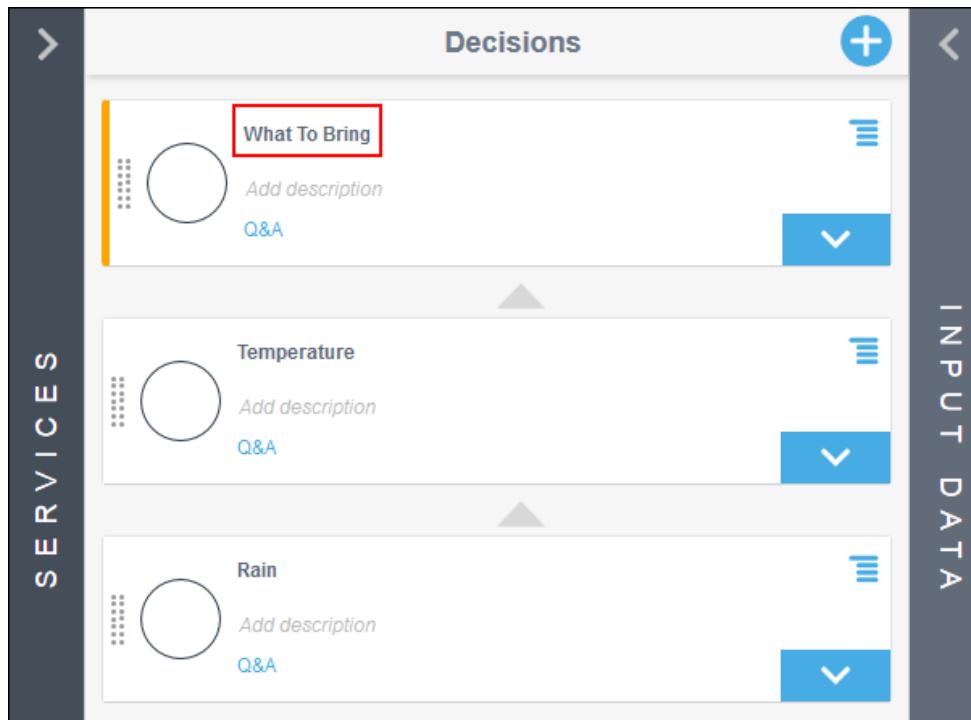
Let's set up our decision model framework. In Decision Model and Notation (DMN), a decision model answers a central question. This model answers the question: what to bring to be prepared for the weather?

1. Add a result decision.
 - a. Click **Add new decision**  in the Decisions bar.
 - b. In the Create Decision dialog box, enter a decision name (*What To Bring*), select **Empty Logic**, and click **Create**. The new decision displays in the Decisions pane. You can change its logic type later.

2. Create two more decisions to act as supporting decisions.

Most decisions models include supporting decisions that feed into the result decision. Deciding what to bring requires supporting decisions to answer these questions: 1) Will it be warm or cold? 2) Will it rain?

- a. Click **Add new decision**  to create another decision. Enter a decision name (*Temperature*), select **Empty Logic**, and click **Create**.
- b. Repeat to create another empty logic decision called **Rain**.



Notice that the topmost decision is the **main result** decision. **Supporting** decisions are listed below the main result decision. Decision order is important.

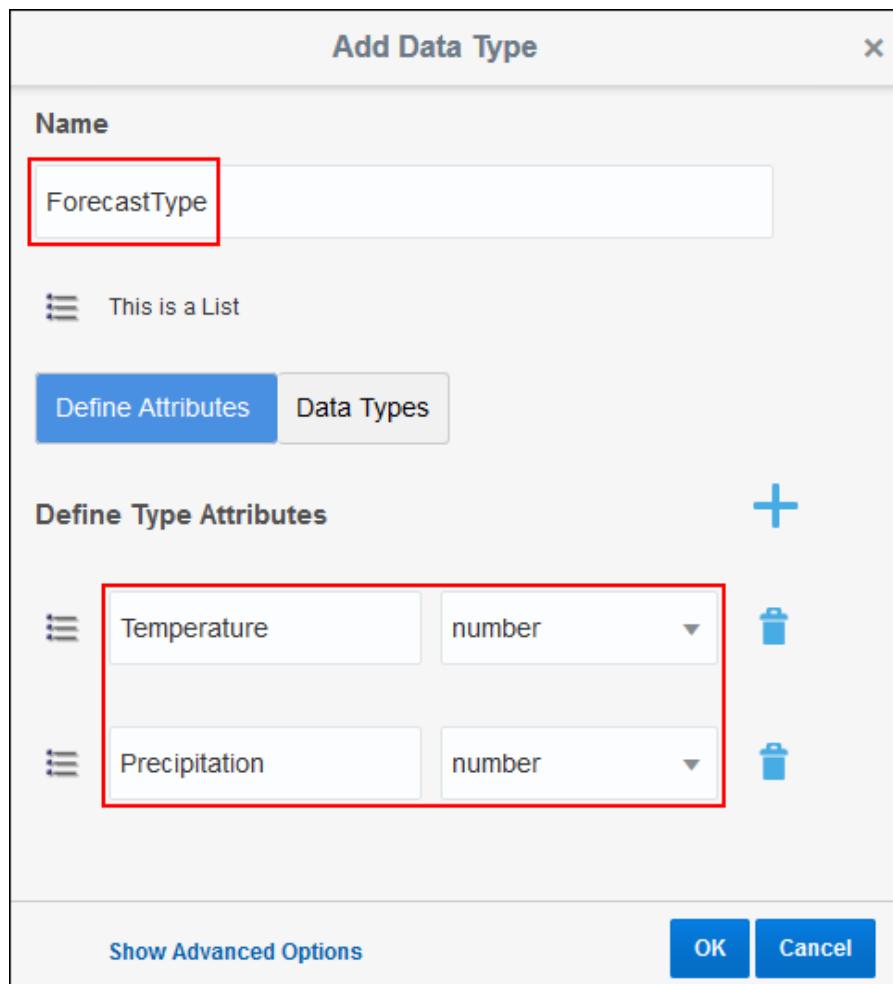
- Decision logic can reference the result of supporting decisions by name. For example, the decision table for **What To Bring** can reference the result of supporting decisions **Rain** and **Temperature** by name.
- Supporting decisions must always be beneath the decisions that reference them. Decision logic can't "see" the decisions listed above. To change the order of decisions in the model, drag the decision by its dotted handle and drop it at the desired position so that supporting decisions are always beneath the decisions that they support.

Define Input

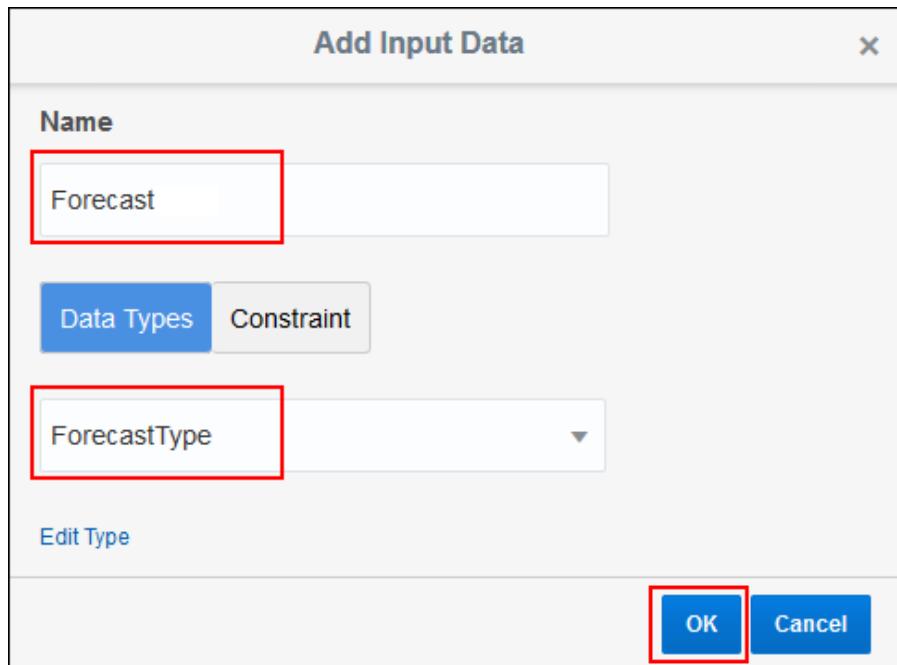
It's hard to make a good decision without input (or data). For our "What to Bring for the Weather" example, the input will be two numbers: temperature and rain percent forecasts. Let's set up their definitions.

- Click **Show data panel** to expand the Input Data pane.
- Add a complex type definition for the forecasts.
 - In the **Type Definition** section, click **Add new item definition**
 - In the **Name** field, enter `ForecastType`.
 - Click **Add Attribute** next to **Define Type Attributes** twice to add two attributes.
 - For the first attribute, enter `Temperature` for its name and select **number** as its type.
 - For the second attribute, enter `Precipitation` for its name and select **number** as its type.

f. Click **OK**.



3. Add an input type that uses the Forecast type definition you just created.
 - a. In the **Input Data** section, click **Add new input data**
 - b. In the **Name** field, enter `Forecast`.
 - c. In the **Type** field, select `ForecastType`.
 - d. Click **OK**.



4. Click **Hide data panel** ➤ to collapse the Input Data pane.

Model Decision Logic

Now let's change the empty logic of the decisions. You can configure decision logic in a variety of ways, but here you'll create a Boolean expression, an if-then-else expression, and a decision table.

1. Configure a Boolean expression for the rain forecast.

- a. Change the **Rain** decision to an Expression decision. Select the **Expression** logic from the list that is displayed within the decision.

Notice that the decision's icon changed to an expression icon and the decision expanded. Also, a number and a yellow circle with an exclamation point **1 !** indicate a validation error because you haven't entered an expression yet.

- b. Enter the expression using the input data you already defined. Click the **Enter expression** field and enter this condition:

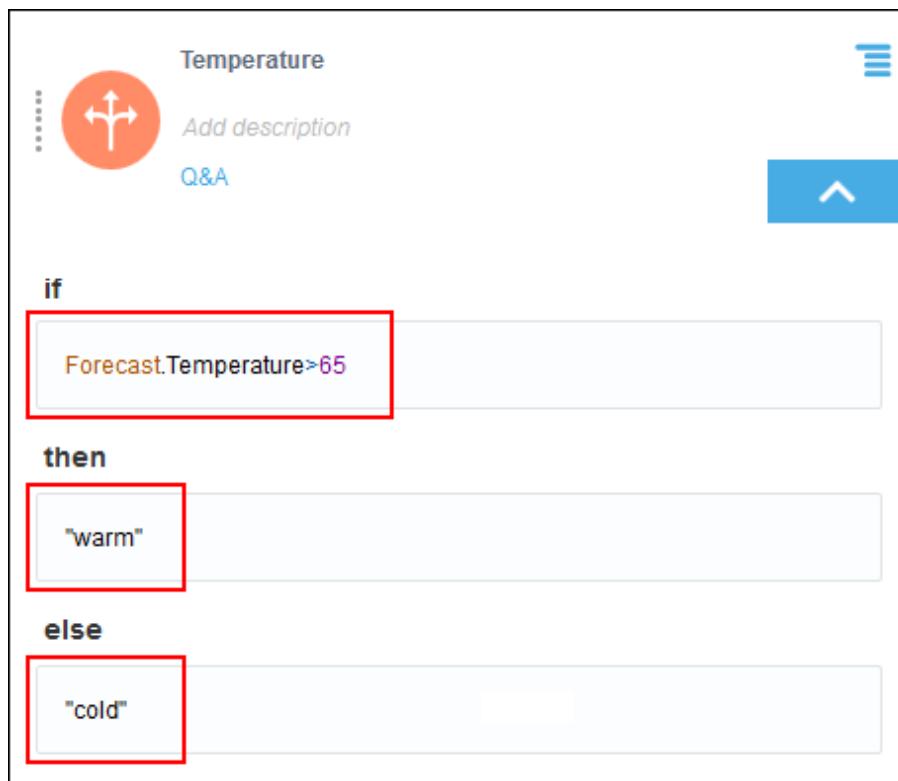
Forecast.Precipitation>80



When the precipitation input value is greater than 80 percent, this Boolean expression will be true (rain is likely); otherwise it will be false (rain is unlikely).

- c. Click **Collapse**  to collapse the Boolean expression. Notice that the validation error no longer displays because the expression is now valid.
2. Configure an if-then-else decision for the temperature forecast.
 - a. Expand the **Temperature** decision and change its type to **If-Then-Else**. The decision expands and shows validation errors for items you need to complete.
 - b. Complete the decision's **if**, **then**, and **else** expression fields as follows:
 - **if:** Forecast.Temperature>65
 - **then:** "warm"
 - **else:** "cold"

Because "warm" and "cold" values are strings, you must include quotation marks around the values. (Depending on your location, you may want to adjust the temperature condition of **>65** from degrees in Fahrenheit to **>18** in Celsius.)



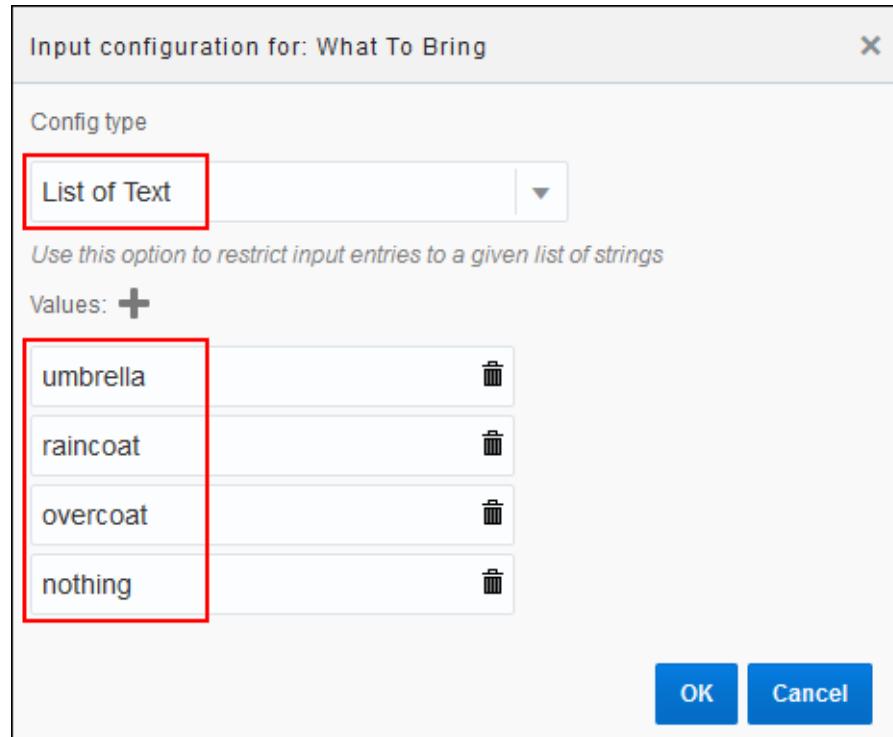
- c. Collapse the decision and verify that the validation errors no longer display.
3. Configure a decision table for the **What To Bring** decision.

A decision table lets you model input, output, and rules in a compact way. It derives a final output decision using input data and the results of other decisions.

In the example, the temperature and rain input decisions each have two possibilities, resulting in a total of four output possibilities.

Temperature (input)	Rain (input)	What To Bring (output)
warm	true	umbrella
cold	true	raincoat
cold	false	overcoat
warm	false	nothing

- Expand the **What To Bring** decision and change its type to **Decision Table**.
- Expand the decision and explore the table started for you.
 - The yellow column is for **output**, and is always the final column of the table. **What To Bring** is already designated as your output decision.
 - The adjacent grey column is for **input**. You'll add a second input column, using one of the controls located above the output column.
 - Rows are for **rules**. A row labeled **1** is already listed.
 - Above the **1** in rule **1** is a **U**, indicating that the table uses a unique hit policy. The hit policy determines the output of a decision table when more than one rule matches the input data.
- Configure the possible values for the **What To Bring** output column.
 - In the **What To Bring** column, click **Enter Allowed Values**.
 - In the **Config type** field, select **List of Text**.
 - Click **Add item** four times to add four values. Enter `umbrella`, `raincoat`, `overcoat`, and `nothing`, and click **OK**.



- Configure the **Temperature** input column.

In the **Enter Expression** cell of the gray input column, press **Ctrl+Space** to open a pop-up menu. Select **Temperature** from the menu. For an input

column, the Allowed Value cell is auto-populated based on the input expression.

- e. Add and configure the **Rain** input column.
 - Click the **What To Bring** output column and select **Add column before** .
 - In the **Enter Expression** cell, press **Ctrl+Space** to open a pop-up menu. Select **Rain** from the menu. The Allowed Value cell is auto-populated with Boolean values.
- f. Add and configure rules.
 - In row 1, click the cell below Temperature and select **warm**.

Notice the  icon in the cell, which stands for text. Clicking this icon lets you switch between entering a string (, an expression (, or date and time values ().

With a text switch mode, text automatically displays italicized and quotation marks aren't needed.
 - Click the cell below Rain and enter **true**. Leave its switch mode as expression because Boolean values are entered as expressions.
 - Click the cell below What To Bring and select **umbrella** from the input values. If needed, set the cell's switch mode to text. Quotation marks aren't needed.
- g. Click **Add rule after**  three times to add three rows to the table.
- h. Complete rules 2 through 4 of the decision table:
 - For rule 2, select **cold**, **true**, and **raincoat**.
 - For rule 3, select **cold**, **false**, and **overcoat**.
 - For rule 4, select **warm**, **false**, and **nothing**.

The screenshot shows the Oracle Decision Modeler interface with a title bar 'Decisions'. Below it is a card titled 'What To Bring' with icons for a calendar, 'Add description', and 'Q&A'. A large button with a blue arrow points upwards. The main area is a 'Decision Table' with the following data:

U	Temperature	Rain	What To Bring
	warm,cold	true, false	umbrella,raincoat,overcoat,nothing
1	warm	true	umbrella
2	cold	true	raincoat
3	cold	false	overcoat
4	warm	false	nothing

- i. Collapse the decision.

Test Your Decisions

Verify that the model produces the results you want by testing it with input values.

1. Click **Test**
2. In the Test Decision Model pane, enter values in the **Temperature** and **Precipitation** fields and click **Start Test**. For example, enter 85 into both fields.

The Decision Model Result pane displays decision results. Click each green check mark to see the decision's result.

The screenshot shows the 'Decision Model Result' pane with a 'Go Back' button. It lists three items with green checkmarks:

What To Bring	Results
Temperature	Output is: umbrella
Rain	

3. Click **Go Back**, and repeat step 1 and 2 to test each decision rule's outcome.

Temperature (F)	Precipitation %	Outcome
85	85	<ul style="list-style-type: none"> • What to Bring: umbrella • Temperature: warm • Rain: true

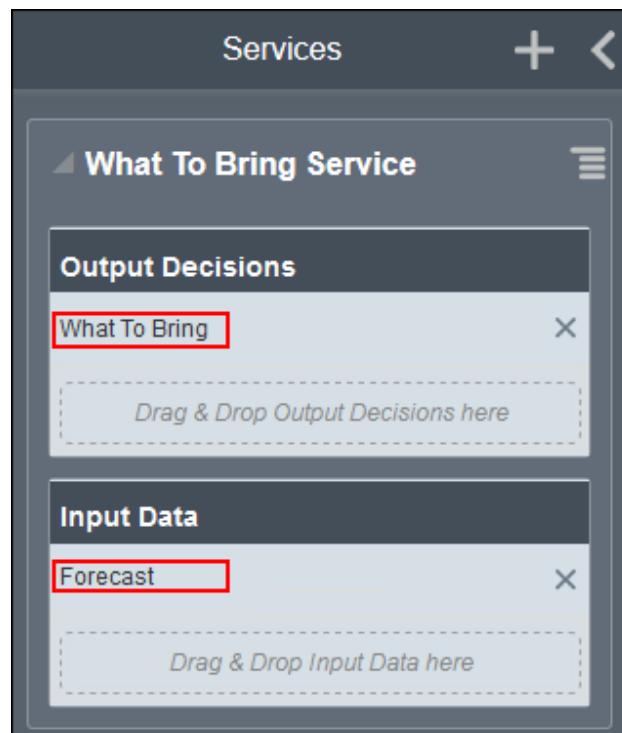
Temperature (F)	Precipitation %	Outcome
50	85	<ul style="list-style-type: none"> • What to Bring: raincoat • Temperature: cold • Rain: true
50	50	<ul style="list-style-type: none"> • What to Bring: overcoat • Temperature: cold • Rain: false
85	50	<ul style="list-style-type: none"> • What to Bring: nothing • Temperature: warm • Rain: false

4. Save the decision model.

Create a Service

After you complete and test the decision logic, create a RESTful service to communicate with Process. The decision service specifies the decision and input data you want to expose.

1. Expand the Services pane, and click **Add new service** .
2. Enter a name for the decision service (**What To Bring Service**) and click **OK**.
3. From the Decisions pane, drag **What To Bring** (the title of your result decision) to the **Output Decisions** field in the Services pane and drop when the field turns blue. The service name displays above the field. Validation errors display because you haven't specified input data yet.
4. From the Input Data pane, drag the **Forecast** input data to the **Input Data** field in the Services pane and drop when the field turns blue.

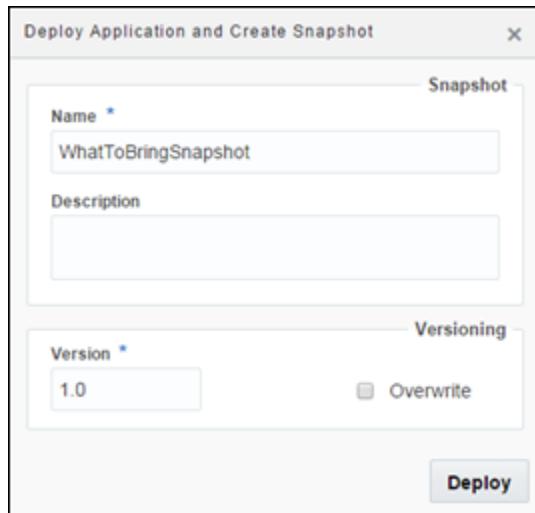


5. Click **Save**.

Deploy a Decision Snapshot

To use a decision model, you must create a snapshot of it and deploy the snapshot on the DMN server.

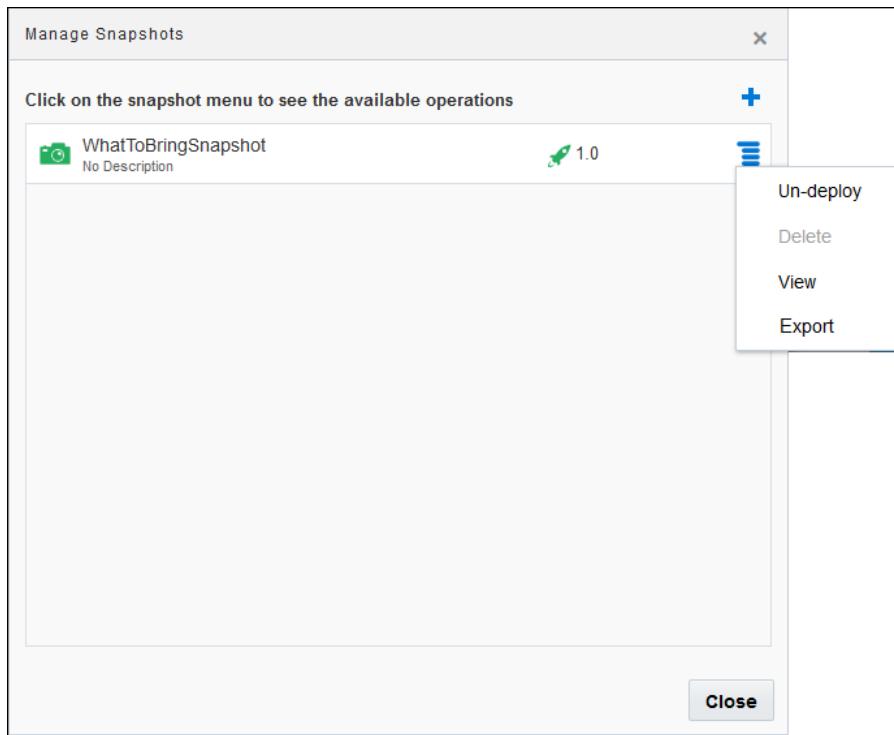
1. In the decision model editor toolbar, click **Deploy**.
2. In the Deploy Application and Create Snapshot dialog box, enter a name (*What to Bring Snapshot*) and click **Deploy**.



3. In the decision model editor toolbar, click **Manage Snapshots**.

Notice that the snapshot you just deployed is listed, along with any other deployed snapshots. Here you can:

- Deploy or undeploy a selected snapshot.
- Delete a snapshot. You must undeploy a snapshot before you can delete it.
- View a read-only version of the snapshot.
- Export a snapshot, which you can later import into a new decision model and edit its contents.



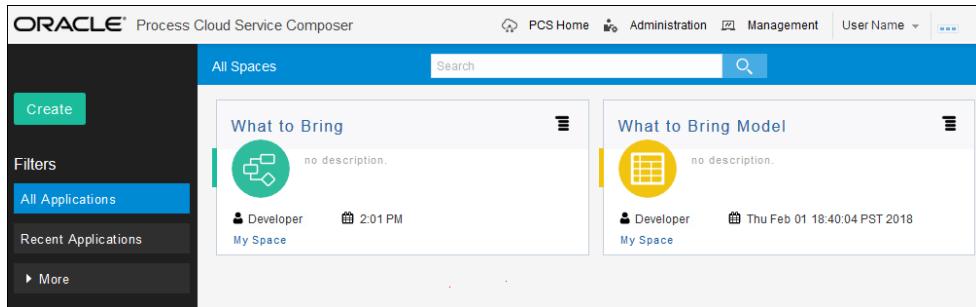
4. Click **Close**.

Use the Decision in Your Application

Now let's use your deployed decision snapshot in a process within an application.

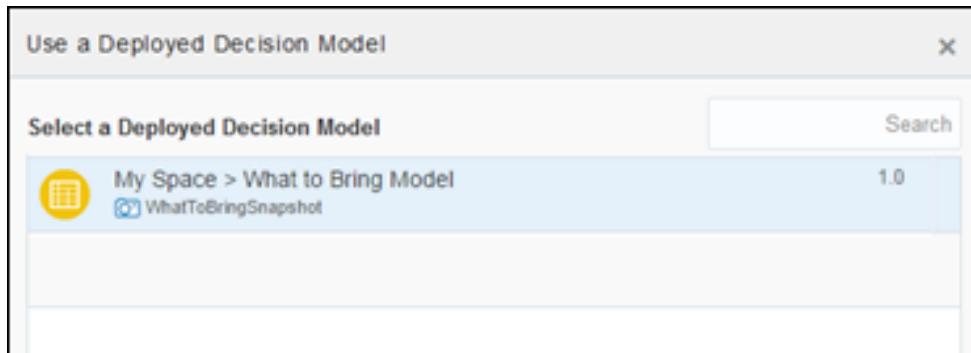
1. Return to the Composer Home page by clicking the Spaces link (**My Space**) in the toolbar.

Notice that the page displays both the application and the decision model you created.

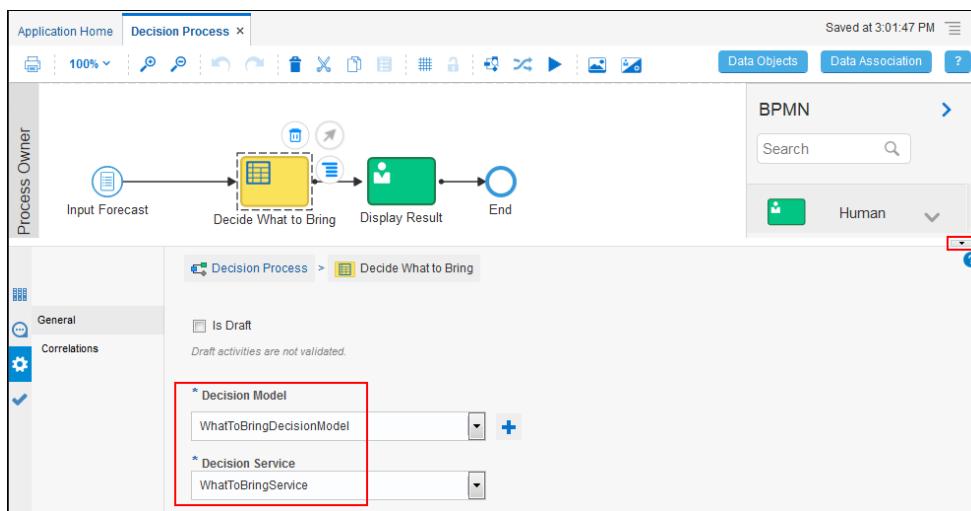


2. Open the **What To Bring** application you created.
3. Open the decision process. Select the decision element you added (**Decide What to Bring**), click **Menu** , and select **Open Properties**.
4. On the Properties pane, click **Use Decision Model**  next to the **Decision Model** field.

- In the Use a Deployed Decision Model dialog box, select the snapshot you created and deployed (`WhatToBringSnapshot`), and click **Use**.



Notice that the Properties pane now lists the decision model and service you selected to use.

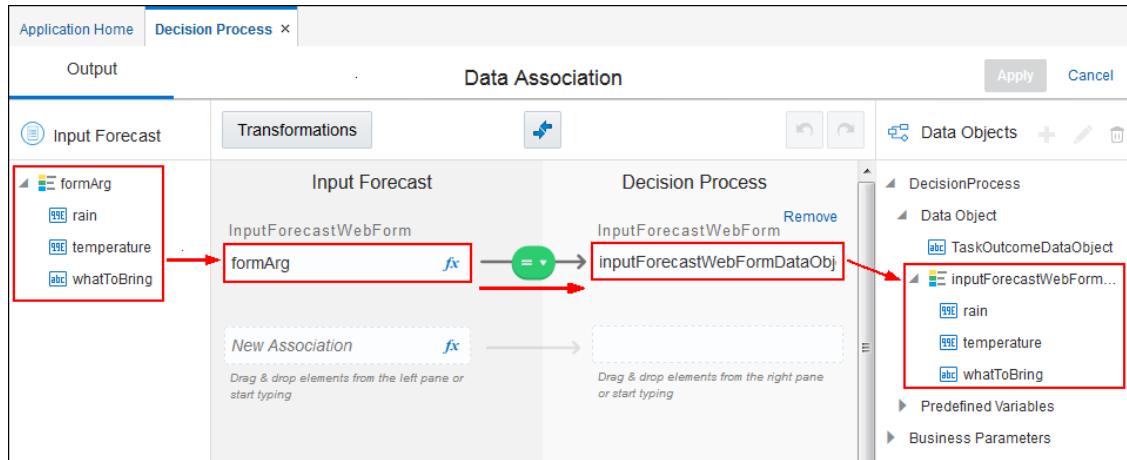


- Click **Collapse Pane** to close the properties pane.

Map Data to and from the Decision

Configuring the data mapping is the last key piece in configuring a decision model. The data needs to flow from the web form into the decision and its result must flow out of the decision so the human task can display it.

- On the Decision Process tab, let's view the data association that was automatically configured for the Input Forecast start activity and the Display Result human task.
 - Select the **Input Forecast** element and click **Data Association** in the toolbar. Expand the values in both side panes. Notice that values entered into the web form are written to a data object called `inputForecastWebFormDataObject`. Click **Cancel**.

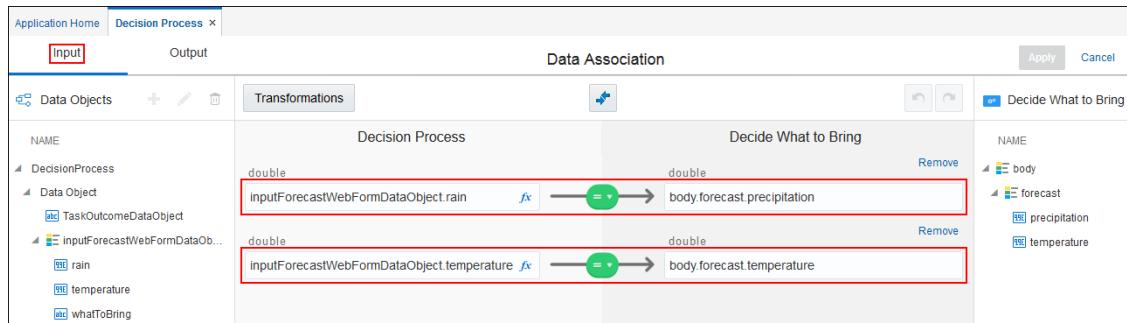


- b. Select the **Display Result** element and click **Data Association** in the toolbar. Notice that for input, the same data object—**inputForecastWebFormDataObject**—is used to write values back to the web form. Click **Cancel**.
2. Map input and output data association for the decision element.

The decision needs to use the input values (rain, temperature) contained in **inputForecastWebFormDataObject** and output the decision's result (what to bring) into the data object.

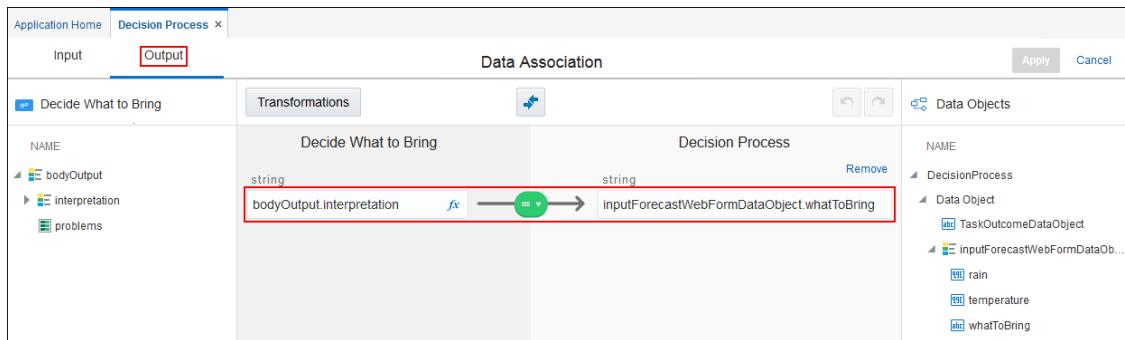
- a. Open data association for the **Decide What to Bring** element. With the **Input** tab selected, expand the values in both side panes.
- b. Map the rain and temperature values as follows:

```
inputForecastWebFormDataObject.rain to body.forecast.precipitation
inputForecastWebFormDataObject.temperature to body.forecast.temperature
```



- c. With the **Output** tab selected, map the **Decide What to Bring** value as follows:

```
bodyOutput.interpretation to inputForecastWebFormDataObject.whattobring
```



3. Click **Apply**.

Try Your Decision in Runtime

Let's try running the application in test mode as an end user.

1. From the toolbar, click **Test**.
2. On the Test Application tab, click **Deploy**. In the Deploy to Test dialog box, leave the **Add Me to All Roles** check box selected and click **Deploy**.
3. After the application deploys successfully, click **Try in Workspace**. When the new browser tab opens displaying runtime options, click **Forecast** from the My Apps pane.
4. In the form, enter temperature and rain values, and click **Submit**.
5. Click **My Tasks** and open your new task. The **What To bring** field displays the decision's result based on the values you entered.

Working with Decision Models

When creating a business process for your application, very often, you'll need to create decisions that enable you to automate policies, computations, and reasoning. Instead of creating multiple decisions, you can create a decision model consisting of

decisions and sub-decisions, and associated decision services to allow you to use the decision model in your business processes. Process allows you to create decision models that make business processes less complex, easier to manage, and more robust in the face of change.

Decision models consist of :

- Decisions and sub-decisions along with the implementation logic
- Input data and type
- Associated decision services

Creating a decision model involves the following main tasks, as described in detail in subsequent sections:

- Create a decision model, the container for decisions, sub-decisions, input data, and decision services. See [Creating a Decision Model](#).
- Add decisions and sub-decisions. See [Adding and Ordering Decisions](#).
- Define the input data and type for your decisions. See [Defining Decision Input and Type](#).
- Model the decision logic. Use Friendly Enough Expression Language (FEEL) to define expressions within all notations of decision logic, including decision tables. FEEL is defined by DMN to provide standard executable semantics to all expressions used within a decision model. See [Understanding FEEL \(Friendly Enough Expression Language\)](#).
- Test the decisions and sub-decisions within your decision model to ensure that they work as expected. See [Testing Decisions](#).
- Create snapshots of your decision model, and deploy these on the DMN server. Decision snapshots are read-only copies of a decision model at a particular moment. These snapshots, once deployed, can be used independently in multiple applications. You can create, delete or deploy snapshots of your decision model or view read-only copies of a snapshot that has already been deployed on the DMN server. See [Deploying and Managing Decision Model Snapshots](#).
- Create decision services to use the deployed decision model snapshots in your Process applications. See [Adding Decisions to Applications and Processes](#).

Decision models facilitate the modeling of complex decisions as a hierarchy of simple decisions. A decision model as a whole, or its decisions and sub-decisions can be used in multiple processes and applications.

You can share decision models with other users directly through the file system. See [Importing and Exporting Decision Models](#).

Creating a Decision Model

Create decision models to make your business processes less complex, easier to manage, and more robust in the face of change.

To create a decision model:

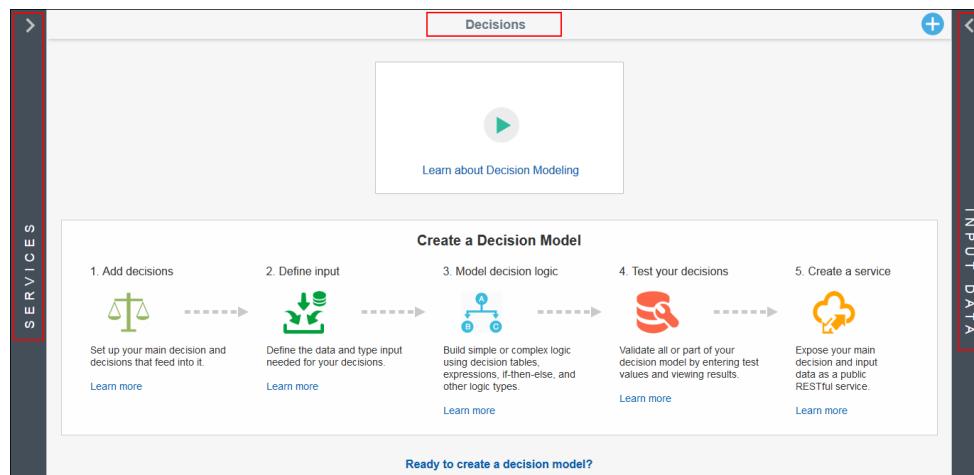
1. On the Composer Home page, click **Create** and select **New Decision Model**.
2. Enter information into the Create Decision Model dialog box. Here are a few things to note:

Field	Description
Name and Description	Make sure the decision model name and description are useful. Give the user a good idea of what the decision model is all about and why they might want to use it. Good names and descriptions help users distinguish decision models with a similar title or purpose. Also, you can't change the name after the decision model is created.
Space	Select the space where the new decision model will be stored. To create a new space, select New Space from the drop-down list.
Open Immediately	Most of the time you'll want to leave the Open Immediately check box selected so you can start configuring and testing your decision model right away. Deselecting the check box is good if you want to create placeholders for several decision models at once and modify them at a later time.

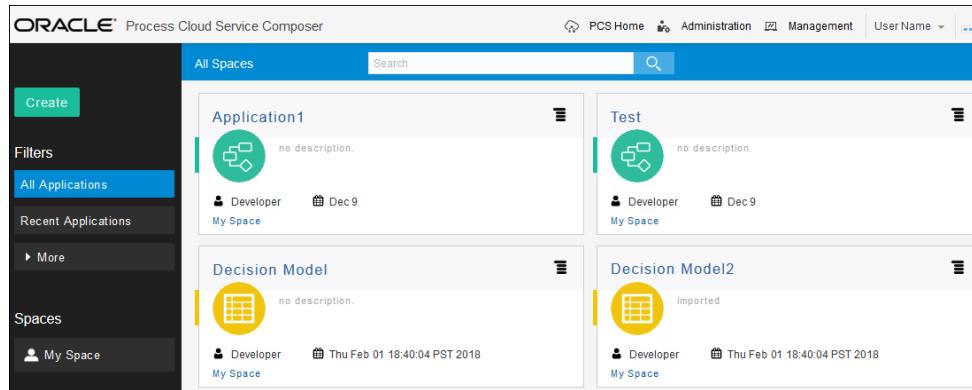
3. Click **Create**.

The decision model editor enables you to:

- Add decisions and sub-decisions
- Define the input data
- Model the decision logic
- Test your decisions
- Create decision services to implement your decision model



You can edit an existing decision model by clicking and opening it in the decision model editor.

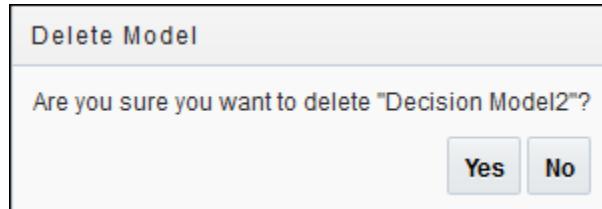


Note:

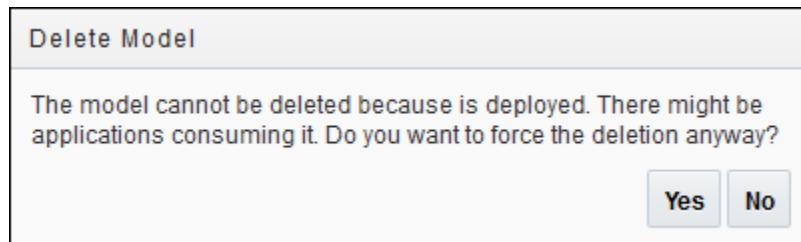
If you add a new service or change the service definition within your decision model, you'll need to create a new reference to the decision model on Process, re-implement the activity, and re-associate the data.

Within the selected decision model, click **Options** of the decision, service, input data, or input type to edit it.

To delete a decision model, click **Options** and then select **Delete Model**.



Deleting a decision model that is currently in use in one or more applications deactivates and deletes all the deployed decision model snapshots.



 **WARNING:**

Deleting a decision model or its snapshot that is currently in use in one or more applications does not delete references to it in the Process application. This may result in errors during runtime when the application tries to call the decision service associated with a decision model that has previously been deleted from the DMN server.

Adding and Ordering Decisions

A decision is a container for logical notations such as decision tables, expressions, if-then rules, and so on. In Process, you create a decision by selecting the logic type you plan to use to obtain a specific output.

Decision models contain:

- Main Decision: The output of the main decision provides the result of the decision model.
- Supporting Decision: The supporting decisions provide input to the main decision.

Decision order is important. The topmost decision is the main decision. Supporting decisions are listed below it. Decision logic can reference the result of supporting decisions by name. Supporting decisions must always be beneath the decision that references them.

To change the order of decisions in the model, drag the decision by its dotted handle and drop it at the desired position so that supporting decisions are always beneath the decisions that they support.

To create a new decision:

1. Click **Add new decision**  in the Decision Model home page to open the Create Decision dialog box.
2. Choose the logic type you want to use, for example, Decision Table. The available logic types are:
 - Empty Logic
 - Decision Table
 - Expression
 - If-Then-Else
 - Function
 - Context
 - Relation
 - List
3. Enter the required information into the Create Decision dialog box.

Field	Description
Name	Enter a unique name for the decision.

Field	Description
Description	Provide additional information, if any, for the decision.
Question	Add the questions allowed for the decision.
Allowed Answers	Add the possible answers for the decision.
Select Output Type	<p>Select the output type for the decision.</p> <p>Set this field if you are certain about the decision output type. If the type selected in this field does not match the actual output type, a validation error occurs. If no selection is made, the decision output type is automatically inferred from its inputs.</p>

4. Click **Create**.

Use the **Options**  menu of a decision to edit or delete it.

Understanding FEEL (Friendly Enough Expression Language)

Decision Modeling and Notation (DMN) defines Friendly Enough Expression Language (FEEL) to provide standard executable semantics to all expressions used within a decision model.

In Process, you use FEEL to define expressions within all notations of decision logic, including decision tables.

Topics

- [Data Types](#)
- [Grammar Rules](#)
- [Built-In Functions](#)
- [Working with Date, Time, and Duration Functions](#)

Data Types

Process supports the following FEEL data types that you can use for input data, expression values, function arguments and return values.

Data Type	Description
Number	FEEL Numbers are based on IEEE 754-2008 Decimal128 format, with 34 decimal digits of precision and rounding toward the nearest neighbor with ties favoring the even neighbor. Numbers are a restriction of the XML Schema type <code>precisionDecimal</code> , and are equivalent to Java <code>BigDecimal</code> with <code>MathContext DECIMAL128</code> .
String	Variable-length sequence of characters italicized or encapsulated in double quotes.
Boolean	Logical Boolean (true/false).
Date and Time	Calendar date and time combination.

 **Note:**

You can extend these basic data types by defining custom data types. See [Defining a Complex Data Type](#).

Grammar Rules

This section provides the syntax for commonly-used FEEL expressions through simple examples. For the complete definition of FEEL syntax, see *Decision Model and Notation (DMN), v1.1*.

Topics

- [Arithmetic Expressions](#)
- [Interval Expressions](#)
- [Comparison Expressions](#)
- [Other Expressions](#)

Arithmetic Expressions

Name	FEEL Expression	Return Value
Addition (+)	0.15+30	30.15
Subtraction (-)	15-30	-15
Multiplication (*)	.20*40.02	8.004
Division (/)	1/50	0.02
Exponentiation (**)	2**3	8

Interval Expressions

Start Value	End Value	FEEL Expression	Return Value
Inclusive	Inclusive	15 in [15..30]	true
Exclusive	Exclusive	15 in (15..30)	false
Exclusive	Inclusive	30 in (15..30]	true
Inclusive	Exclusive	30 in [15..30)	false

 **Note:**

In decision tablet input entry and input/output allowed value cells, you can use intervals or list of intervals to test against the input data.

Comparison Expressions

Name	FEEL Expression	Return Value
Less than (<)	8<2**3	false

Name	FEEL Expression	Return Value
Less than or equal to (\leq)	15 in (≤ 15)	true
Equal ($=$)	8=2**3	true
Greater than ($>$)	30 in (> 30)	false
Greater than or equal to (\geq)	1/5>=0.20	true
Not equal to (\neq)	8!=2**3	false

 **Note:**

In decision table input entry and input/output allowed value cells, you can use comparison operators to define unary expressions.

Other Expressions

Name	FEEL Expression	Return Value
Disjunction	($2*2=2^{**}2$) or ($3*2=3^{**}2$)	true
Conjunction	($2*2=2^{**}2$) and ($3*2=3^{**}2$)	false
Negation	not($2*2=2^{**}2$)	false

 **Note:**

In decision tablet input entry and input/output allowed value cells, you can use comma-separated list of values to specify disjunction.

Built-In Functions

FEEL includes a library of built-in functions that you can use to define expressions.

Process supports the following types of built-in functions:

- [Conversion Functions](#)
- [Boolean Functions](#)
- [String Functions](#)
- [List Functions](#)
- [Numeric Functions](#)

Conversion Functions

Name(parameters)	Parameter Domain	Description	Example
date(<i>from</i>)	date string	convert <i>from</i> to a date	date("2012-12-25") – date("2012-12-24") = duration("P1D")

Name(parameters)	Parameter Domain	Description	Example
date(<i>from</i>)	date and time	convert <i>from</i> to a date (set time components to null)	date(date and time("2012-12-25T11:00:00Z")) = date("2012-12-25")
date and time(<i>from</i>)	date time string	convert <i>from</i> to a date and time	date and time("2012-12-24T23:59:00") + duration("PT1M") = date and time("2012-12-25T00:00:00")
time(<i>from</i>)	time string	convert <i>from</i> to time	time("23:59:00z") + duration("PT2M") = time("00:01:00@Etc/UT C")
time(<i>from</i>)	time, date and time	convert <i>from</i> to time (ignoring date components)	time(date and time("2012-12-25T11:00:00Z")) = time("11:00:00Z")
number(<i>from</i> , grouping separator, decimal separator)	<i>hour</i> , <i>minute</i> , <i>second</i> , are numbers, offset is a days and time duration, or null string1, string, string	convert <i>from</i> to a number	number("1 000,0", ",",".") = number("1,000.0", ",",".")
string(<i>from</i>)	non-null	convert <i>from</i> to a string	string(1.1) = "1.1" string(null) = null
duration(<i>from</i>)	duration string	convert <i>from</i> to a days and time or years and months duration	<ul style="list-style-type: none"> • date and time("2012-12-24T23:59:00") - date and time("2012-12-22T03:45:00") = duration("P2DT20H14M") • duration("P2Y2M") = duration("P26M")
years and months duration(<i>from</i> , <i>to</i>)	both are date and time	return years and months duration between <i>from</i> and <i>to</i>	years and months duration(date("2011-12-22"), date("2013-08-24")) = duration("P1Y8M")

Boolean Functions

Name(parameters)	Parameter Domain	Description	Example
not(<i>negand</i>)	boolean	logical negation	<ul style="list-style-type: none"> • not(true) = false • not(null) = null

String Functions

Name(parameters)	Parameter Domain	Description	Example
substring(<i>string</i> , <i>start position</i> , <i>length?</i>)	string, number1	return <i>length</i> (or all) characters in <i>string</i> , starting at <i>start position</i> . 1 st position is 1, last position is -1	<ul style="list-style-type: none"> • substring("foobar",3) = "obar" • substring("foobar",3,3) = "oba" • substring("foobar", -2, 1) = "a"
string length(<i>string</i>)	string	return length of <i>string</i>	string length("foo") = 3

Name(parameters)	Parameter Domain	Description	Example
upper case(<i>string</i>)	string	return uppercased <i>string</i>	upper case("aBc4") = "ABC4"
lower case(<i>string</i>)	string	return lowercased <i>string</i>	lower case("aBc4") = "abc4"
substring before (<i>string</i> , <i>match</i>)	string, string	return substring of <i>string</i> before the <i>match</i> in <i>string</i>	<ul style="list-style-type: none"> • substring before("foobar", "bar") = "foo" • substring before("foobar", "xyz") = ""
substring after (<i>string</i> , <i>match</i>)	string, string	return substring of <i>string</i> after the <i>match</i> in <i>string</i>	<ul style="list-style-type: none"> • substring after("foobar", "ob") = "ar" • substring after("", "a") = ""
replace(<i>input</i> , <i>pattern</i> , <i>replacement</i> , <i>flags?</i>)	string2	regular expression pattern matching and replacement	replace("abcd", "(ab) (a)", "[1=\$1] [2=\$2]") = "[1=ab] [2=]cd"
contains(<i>string</i> , <i>match</i>)	string	does the <i>string</i> contain the <i>match</i> ?	contains("foobar", "of") = false
starts with(<i>string</i> , <i>match</i>)	string	does the <i>string</i> start with the <i>match</i> ?	starts with("foobar", "fo") = true
ends with(<i>string</i> , <i>match</i>)	string	does the <i>string</i> end with the <i>match</i> ?	ends with("foobar", "r") = true
matches(<i>input</i> , <i>pattern</i> , <i>flags?</i>)	string2	does the <i>input</i> match the regexp <i>pattern</i> ?	matches("foobar", "^fo*b") = true

List Functions

Name(parameters)	Parameter Domain	Description	Example
list contains(<i>list</i> , <i>element</i>)	list, any element of the semantic domain including null	does the <i>list</i> contain the <i>element</i> ?	list contains([1,2,3], 2) = true
count(<i>list</i>)	list	return size of <i>list</i>	count([1,2,3]) = 3
minimum(<i>list</i>)	(list of) comparable items	return minimum item	minimum([1,2,3]) = 1
maximum(<i>list</i>)	(list of) comparable items	return maximum item	maximum([1,2,3]) = 3
sum(<i>list</i>)	(list of) numbers	return sum of numbers	sum([1,2,3]) = 6
mean(<i>list</i>)	(list of) numbers	return arithmetic mean (average) of numbers	mean([1,2,3]) = 2
sublist(<i>list</i> , <i>start position</i> , <i>length?</i>)	list, number1, number2	return list of <i>length</i> (or all) elements of <i>list</i> , starting with <i>list[start position]</i> . 1 st position is 1, last position is -1	sublist([1,2,3], 1, 2) = [2]
append(<i>list</i> , <i>item...</i>)	list, any element including null	return new <i>list</i> with <i>items</i> appended	append([1], 2, 3) = [1,2,3]
concatenate(<i>list...</i>)	list	return new <i>list</i> that is a concatenation of the arguments	concatenate([1,2],[3]) = [1,2,3]

Name(parameters)	Parameter Domain	Description	Example
insert before(<i>list</i> , <i>position</i> , <i>newItem</i>)	list, number1, any element including null	return new <i>list</i> with <i>newItem</i> inserted at <i>position</i>	insert before([1,3],1,2) = [1,2,3]
remove(<i>list</i> , <i>position</i>)	list, number1	<i>list</i> with item at <i>position</i> removed	remove([1,2,3], 2) = [1,3]
reverse(<i>list</i>)	list	reverse the <i>list</i>	reverse([1,2,3]) = [3,2,1]
index of(<i>list</i> , <i>match</i>)	list, any element including null	return ascending list of <i>list</i> positions containing <i>match</i>	index of([1,2,3,2],2) = [2,4]
union(<i>list</i> ...)	list	concatenate with duplicate removal	union([1,2],[2,3]) = [1,2,3]
distinct values(<i>list</i>)	list	duplicate removal	distinct values([1,2,3,2,1]) = [1,2,3]
flatten(<i>list</i>)	list	flatten nested lists	flatten([[1,2],[[3]], 4]) = [1,2,3,4]

List Iterations

Name(parameters)	Description	Example
for [item] in [<i>list</i>] return [expression]	iterate over a list	for <i>i</i> in [1,2,3,4] return <i>i</i> * <i>i</i> = [1,4,9,16]
sum (for [item] in [<i>list</i>] return [expression])	iterate over a list and return the sum of iterations	sum(for <i>i</i> in [1,2,3,4] return <i>i</i> * <i>i</i>) = 30

Numeric Functions

Name(parameters)	Parameter Domain	Description	Example
decimal(<i>n</i> , <i>scale</i>)	number, number1	return <i>n</i> with given <i>scale</i>	<ul style="list-style-type: none"> • decimal(1/3, 2) = .33 • decimal(1.5, 0) = 2 • decimal(2.5, 0) = 2
floor(<i>n</i>)	number	return greatest integer <= <i>n</i>	<ul style="list-style-type: none"> • floor(1.5) = 1 • floor(-1.5) = -2
ceiling(<i>n</i>)	number	return smallest integer >= <i>n</i>	<ul style="list-style-type: none"> • ceiling(1.5) = 2 • ceiling(-1.5) = -1

Working with Date, Time, and Duration Functions

Because FEEL does not support literal representation of date, time, or duration values, you must use a combination of built-in functions and string/number literals to express these values.

Built-in functions extract date, time, and duration data from strings or numbers. The following sections provide examples of a few frequently-used date and time operations using built-in functions:

- [Conversion Operations](#)
- [Arithmetic Operations](#)
- [Comparison Operations](#)

Conversion Operations

Examples in this section demonstrate conversion of number or string literals into date, time, or duration data types.

Date and Time Data Type Examples

The following examples convert string literals to date or date-time values:

Example	Description
date("2012-12-25")	Represents the date 2012-12-25 in the YYYY-MM-DD format
time("T23:59:00")	Represents the time 23:59:00 in the hh:mm:ss format
date and time("2012-12-25T11:00:00")	Represents the date 2012-12-25 and time 11:00:00 in YYYY-MM-DD and hh:mm:ss formats respectively
date and time("2012-12-25T11:00:00Z")	Represents the date 2012-12-25 and time 11:00:00 in YYYY-MM-DD and hh:mm:ss formats respectively; "Z" represents UTC time

The following examples convert number literals to date or date-time values:

Example	Description
date(2015,12,12)	Represents the date 2012-12-12 in the YYYY-MM-DD format
time(23, 59, 0)	Represents the time 23:59:00 in the hh:mm:ss format
date and time(date(2015,12,12),time(23, 59, 0))	Represents the date 2012-12-12 and time 23:59:00 in YYYY-MM-DD and hh:mm:ss formats respectively

Duration Data Type Examples

A few conversion examples using the duration function are listed in the following table:

Example	Description
duration(P1DT12H30M)	Represents a duration of 1 day, 12 hours, and 30 minutes
duration(-P120D)	Represents a duration of minus 120 days
duration(PT2000H)	Represents a duration of 2000 hours
duration(P1Y2M3DT10H30M)	Represents a duration of 1 year, 2 months, 3 days, 10 hours, and 30 minutes

Arithmetic Operations

Examples in this section demonstrate arithmetic operations that can be performed on date, time, or duration data types.

Operator	Example	Result	Description
(+)	date("2016-12-25") + duration("P3Y2M6D")	date("2020-03-02")	Returns the date 3 years 2 months and 6 days after 2016-12-25

Operator	Example	Result	Description
(+)	date and time("2016-12-25T12:30:00Z") + duration("P1Y2M3DT10H30M")	date and time("2018-02-28T23:00:00Z")	Returns the date and time 1 year 2 months 3 days and 10 hours 30 minutes after the date 2016-12-25 and time 12:30:00
(+)	date("2016-12-25") + duration("-P3Y2M6D")	date("2013-10-19")	Returns the date 3 years 2 months and 6 days before 2016-12-25
(-)	date("2015-12-25") - date("2012-12-25")	duration("P1095DT0H0M0S")	Returns a duration indicating number of days and time
(-)	date and time("2012-12-25T12:00:00") - date and time("2015-12-25T11:12:00")	duration("-P1094DT23H12M0S")	Returns a duration indicating number of days and time
(-)	date("2016-12-25") - duration("P3Y2M6D")	date("2013-10-19")	Returns the date 3 years 2 months and 6 days before 2016-12-25
(/)	(date("2015-12-25") - date("2012-12-25"))/duration("P1D")	1095	Returns the number of days between two dates
(/)	(date and time("2015-12-25T17:00:00") - date and time("2015-12-25T09:12:00"))/duration("PT1H")	7.8	Returns the number of hours between two date and time values
(/)	(date("2015-12-25") - date("2015-12-24"))/duration("P1Y")	0.0027397260273972603	Returns the number of years between two dates
None	years and months duration(date("2012-12-23") , date("2015-12-25"))	duration("-P3Y0M")	Returns the years and months duration between two dates

Comparison Operations

Examples in this section demonstrate comparison operations that can be performed on date or time data types.

Operator	Example	Result	Description
>	date("2012-12-25") > date("2015-12-25")	false	Determines if Date A occurs after Date B
>	((date("2015-12-25") - date("2015-11-25")) > duration("P1Y"))	false	Determines if Duration A is greater than Duration B

Defining Decision Input and Type

An **input data** is a placeholder for information to be supplied to decisions when a containing decision service is invoked. Supported data types for input data are string, number, boolean, and datetime.

You can extend the built-in data types to define custom complex data types. See [Data Types](#).

Topics

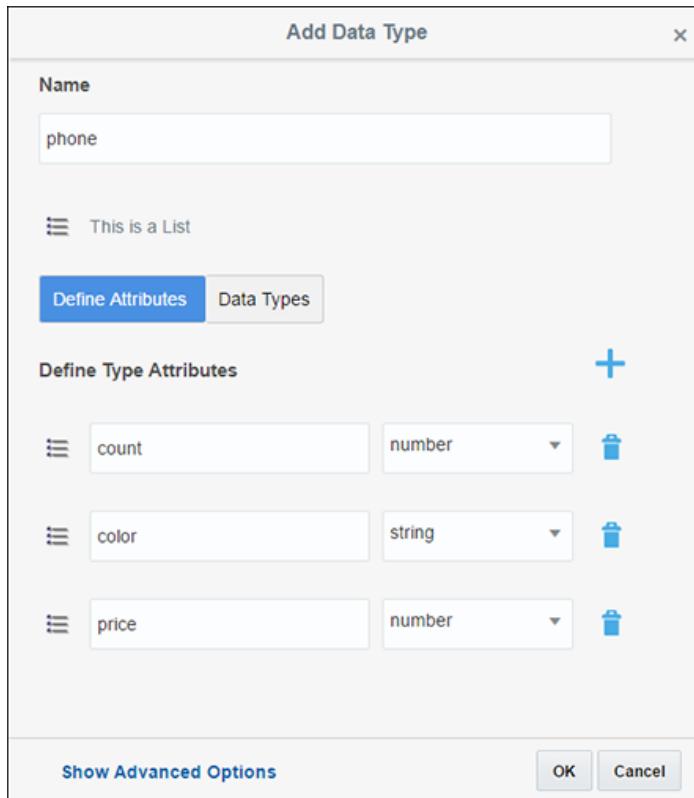
- Defining a Complex Data Type
- Creating Input Data

Defining a Complex Data Type

If the built-in data types aren't suitable for an input variable in your decision model, you can create a complex data type. A **complex data type** is a composite of built-in data types or other complex data types.

To define a new data type:

1. In the Input Data pane, click **Add new item definition**  to open the Add Data Type dialog box.



2. Enter a suitable name for the new data type definition.
 - To identify the data type as a list, click **This is a List** .
 - To use an existing data type for the definition, click **Data Types** and choose an available type from the **Select type** drop-down list.
 - To show the additional Constraint option, click the **Show Advanced Options** link. Click **Constraint** to restrict the data type definition to enumerated values or ranges.
 - To define a new attribute, click **Define Type Attributes** .

 **Note:**

If the **Show Advanced Options** link is selected, you can define two types of attributes using the **Define Type Attributes** (+) button, that is, an attribute of an existing data type or a constraint attribute. Use the **Constraint** option to define an attribute with enumerated values or ranges.

To identify an attribute as a list, click **List**  next to the attribute.

Enter a suitable name, and select an existing data type or enter allowed values or a range for the attribute.

To delete an attribute, click **Delete** .

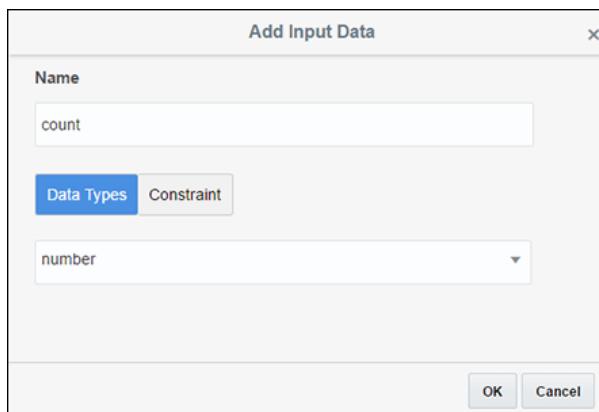
3. Click **OK** to save the data type definition.

To edit or delete a data type definition, click **Menu** .

Creating Input Data

To create a new input data variable:

1. In the **Input Data** section, click **Add new input data** (+) to open the Add Input Data window.



2. Enter a suitable name for the variable. The **Data Types** button is selected by default.
3. Select an existing data type for the variable from the **Select type** drop-down menu or select **[New Type]** and use the **Define New Type** link to define a new complex data type.

To restrict the input data variable to enumerated values or ranges, select the **Constraints** button and enter a list of allowed values or an allowed range.

4. Click **OK** to save the variable.

The **Menu** () icon of an input data variable provides options to edit, delete, or move the variable into a selected service.

Modeling Decision Logic

Model the decision logic by defining how each decision's output is derived from its inputs.

You can use a top-down approach to define the logic within a decision model.

1. To begin with, edit the final output decision, which provides the result of the decision model.
2. Choose a suitable notation to obtain the required result, and configure logic to it. You can choose from several options available, such as a decision table, If-Else expression, context, and so on.
3. Proceed to the next level in the decision framework, and define the logic for each sub-decision.
4. You can use input data and results of other decisions while configuring the logic of a decision.
5. Finalize the decision model by iteratively fine-tuning the decision framework and the logic within each decision.

Process provides following notations to model the logic within a decision:

- [Using Empty Logic Decisions](#)
- [Using Decision Tables](#)
- [Using Expressions](#)
- [Using If-Then-Else Statements](#)
- [Using Functions](#)
- [Using Contexts](#)
- [Using Lists](#)
- [Using Relations](#)

Using Empty Logic Decisions

While outlining the framework of a decision model, you can create a decision with the Empty Logic notation as a placeholder for actual logic.

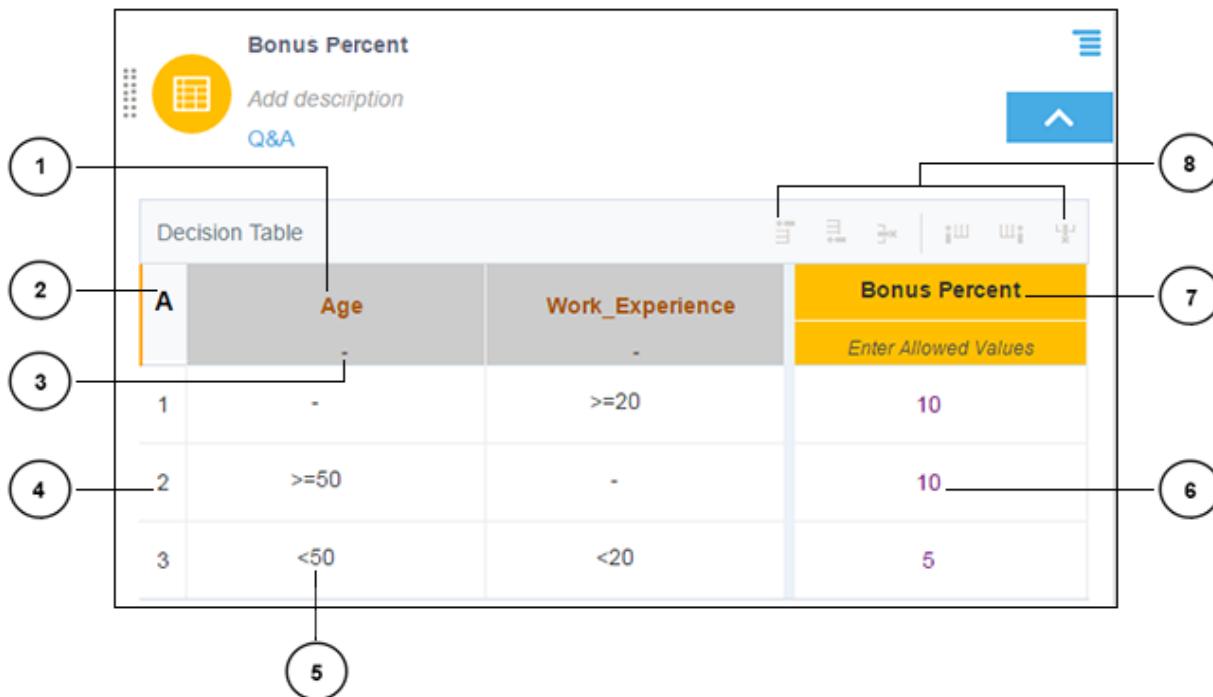
To create a decision with the Empty Logic notation, **Add new decision**  in the Decisions bar and select **Empty Logic** from the Create Decision window.

Using Decision Tables

Decision tables are the notation of choice to model complex logic. The tabular layout of decision tables helps you effectively document all the possible conditions and results of a decision.

To create a decision table, click **Add new decision**  in the Decisions bar and select **Decision Table** from the Create Decision window. An empty decision table appears.

The following figure shows a sample decision table with all its elements noted:



1. Input Expression cell – Contains the expression associated with a particular input column or clause.
 2. Hit Policy Indicator cell – Displays the hit policy selected for the table.
 3. Allowed Values cell – Displays permitted values for cell entries of that column.
 4. Rules – Rows within the table.
 5. Input Entry cell – Contains an input entry.
 6. Output Entry cell – Contains an output entry.
 7. Output Label cell – Contains the name of the decision.
 8. Row and column controls – Controls to add or remove rows and columns.

The following sections explain each decision table element in detail:

- Defining Decision Table Input
 - Defining Decision Table Output
 - Configuring Rules
 - Configuring a Hit Policy

Defining Decision Table Input

An input (also referred to as input clause) to a decision table consists of an input expression and several input entries. It is represented as a column within a table. A decision table may have multiple inputs.

Using Input Expressions

In a decision table, you enter an input expression in the header cell of an input column. In combination with input entries, an input expression determines the value of a particular input column. It can be a simple test expression, for example, Age>50. You can use input variables, outputs of other decisions, or built-in functions to define input expressions.

In an Input Expression cell, press **Ctrl+Space** to open a pop-up menu. Use the options available in the menu to define your expression.

In the Decision Model editor, the expression language used for all expressions, including input expressions, is friendly enough expression language (FEEL). Want to learn more about the expression language syntax? See [Understanding FEEL](#).

Allowed Values

Click on the Allowed Values cell within an Input Expression or Output Label cell to specify permitted values for cell entries of that column. Select from the following available options:

Allowed Value Type	Description
Auto	The DMN server determines allowed entries for the column based on the input expression. This is the default selection for a new input column.
Any	Use this option to specify that there are no restrictions on entries. This is the default selection for the output column.
List of Text	Use this option to restrict entries to a given list of strings.
List of Date and Time	Use this option to restrict entries to a given list of date and time values.
Constraint	Use this option to restrict entries to a given list of values, limit, or range.
Boolean	Use this option to permit only Boolean entries.

Note:

The data type of input entry cells is determined by the data type of the input expression. Make sure that the type of allowed values you supply is consistent with the data type of the input clause.

Using Input Entries

You can enter strings, expressions, or date and time values as input entries based on the mode you select. Depending on the mode and allowed values, an auto-suggest menu appears when you click on an input entry cell.

 **Note:**

If the data type of an input entry does not match the data type of the column, or if the input entry is not among the allowed values, an error is displayed within the decision.

Switch Mode

Within each input entry cell, you can switch between entering a string, an expression, or date and time values. The following table describes all available modes in detail:

Mode	Description
String ( abc)	Use this mode to enter a string. In this mode, you can enter a string as a plain literal without double quotes.
Expression ( fx)	<p>Use this mode to enter an expression. When you switch to this mode, the Unary Test Editor icon () appears at the start of the cell. You can use the options available in the editor to define an expression.</p> <p>Expressions supported in input entry cells are unary expressions, for example, an entry of <code><=15</code>.</p> <p>Want to learn more about the syntax and examples? See Grammar Rules.</p> <p>Note: Boolean values, numbers, and irrelevant (-) entries must be entered as expressions.</p>
DateTime ( date)	Use this mode to enter date and time values.

 **Note:**

Within an input entry cell, you can toggle between String and Expression modes or between DateTime and Expression modes based on the data type of the input expression. If the Input Expression cell is empty, only String and Expression modes are available.

Defining Decision Table Output

An output (also referred to as output clause) of a decision table consists of an output label and several output entries. It constitutes the final column of a table.

Using Output Labels

Generally, the output label is the name of the decision table. In the Output Label cell, you can specify the allowed values for output entries. See Allowed Values in [Defining Decision Table Input](#).

Using Output Entries

You can enter strings, expressions, or date and time values as output entries based on the mode you select. See Switch Mode in [Defining Decision Table Input](#).

Depending on the mode and allowed values, an auto-suggest menu appears when you click on an output entry cell.

Expressions for output entries can be simple expressions defined using input variables, outputs of other decisions, or built-in functions.

Configuring Rules

Rules are expressed as rows within a table. Every rule consists of one or more input entries and a corresponding output entry.

Generally, a decision table consists of multiple rules. When the input data matches the input entries of a rule, the result of the decision table contains the output entry of the rule.

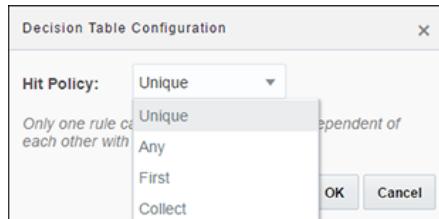
Configuring a Hit Policy

The Hit Policy Indicator cell displays the hit policy selected for the table. The hit policy determines the output of a decision table from the output cells of matched rules. A rule is matched when all of its condition cells match the inputs of the decision table.

Based on the hit policy, Decision Model Notation (DMN) broadly groups decision tables into the following categories:

1. **Single Hit:** A single-hit table returns the output of only one rule. Under the single-hit category, Process supports the following hit policies:
 - **Unique (U)** – Only one of the rules can match.
 - **Any (A)** – Multiple rules can match, but all matching rules must have the same output.
 - **First (F)** – Multiple rules can match; the output of the first rule that matches is returned.
 - **Priority (P)** – Multiple rules can match; the output value that has the highest priority is returned.
2. **Multiple Hit:** A multiple-hit table returns the output of multiple rules. Under this category, the following hit policy is supported in Process:
 - **Collect (C)** – Multiple rules can match; outputs are returned as an arbitrarily-ordered list.
 - **Collect Sum (C+)** – Multiple rules can match; the sum of outputs is returned.
 - **Collect Min (C<)** – Multiple rules can match; the smallest output value is returned.
 - **Collect Max (C>)** – Multiple rules can match; the largest output value is returned.
 - **Collect Count (C#)** – Multiple rules can match; the count is returned.

When you create a new table, the Unique (U) hit policy is selected by default. To change the policy, click the Hit Policy Indicator cell and choose from the available options in the Hit Policy drop-down list. If rules within the table violate the selected hit policy, a warning is displayed within the decision.



Hit Policy Examples

Here are examples for all the hit policies.

- **Single Hit Unique**

In a decision table with Unique hit policy, only one rule can match. All rules are independent of each other, and no overlap is permitted. The decision table returns the output of the rule that matches.

Here is a decision table created with the Unique hit policy:

Decision Table		
U	Temperature	What to Wear
1	<25	Wool coat
2	25	Jacket
3	>25	Casuals

In this example, for any input value of temperature, only one rule can match.

- **Single Hit Any**

In a decision table with Any hit policy, multiple rules can match. The overlap is permitted only if the matching rules have the same output. The decision table returns the output of any one of the matching rules. The hit policy is breached if matching rules have different outputs.

Here is a decision table created with the Any hit policy:

Bonus Percent

Add description
Q&A

Decision Table

A	Age	Work_Experience	Bonus Percent
<i>Enter Allowed Values</i>			
1	-	≥ 20	10
2	≥ 50	-	10
3	< 50	< 20	5

In this example, for an input age of 50 and work experience of 20, the first and second rules match. This overlap is allowed because these rules have the same output. The decision table returns the output of any one of these rules.

- **Single Hit First**

In a decision table with First hit policy, multiple rules with different output entries can match. The output of the lowest-numbered matching rule is the result of the table.

Here is a decision table created with the First hit policy:

Vacation Days

Add description
Q&A

Decision Table

F	Service_Years	Vacation Days
<i>Enter Allowed Values</i>		
1	< 5	5
2	≥ 5	10
3	> 10	15

In this example, if service years are 11, the second and third rules match. The decision table returns only the second rule's output.

- **Single Hit Priority**

In a decision table with Priority hit policy, multiple rules with different output entries can match. The priority of output values (in descending order) is specified as a list in the Allowed Values cell of the output column. The decision table returns the output value that has the highest priority among outputs of all matching rules.

Here is a decision table created with the Priority hit policy:

The screenshot shows a decision table titled "Decision Table". The columns are labeled "P" (Priority), "Age", and "Discount Percent". There are four rows. Row 1 has a priority of 1 and an age range of <18, with a discount of 15. Row 2 has a priority of 2 and an age range of [18..45], with a discount of 5. Row 3 has a priority of 3 and an age range of >45, with a discount of 10. Row 4 has a priority of 4 and an age range of >60, with a discount of 15. The "Discount Percent" column has a header "5, 15, 10" indicating the allowed values in priority order.

P	Age	Discount Percent
1	<18	15
2	[18..45]	5
3	>45	10
4	>60	15

In this example, the last two rules match for an input age of 61. The decision table returns the output value that has the highest priority among of these rules, that is, 15; the priority order is defined in the Allowed Values cell.

- **Multiple Hit Collect**

In a decision table with Collect hit policy, multiple rules with different output entries can match. The decision table returns outputs of all matching rules in an arbitrarily-ordered list.

Here is a decision table created with the Collect hit policy:

The screenshot shows a decision table titled "Interest Rates". The table has three columns: "C" (Category), "Age", and "Interest Rates". The "Age" column contains three rows: "1 >18", "2 >60", and "3 <18". The "Interest Rates" column contains the values "10", "8", and "5" respectively. The table is set against a background of a mobile application interface with a navigation bar at the top.

C	Age	Interest Rates
1	>18	10
2	>60	8
3	<18	5

In this example, two rules match for an input age of 61. The decision table returns output values of these rules in a list:

The screenshot shows a results list titled "Results". It contains two items: "10" and "8", each in its own row. The list is part of a larger interface with a header and other sections visible.

Results
10
8

- **Multiple Hit Collect (Sum)**

In a decision table with Collect (Sum) hit policy, multiple rules with different output entries can match. The decision table returns the sum of outputs of all matching rules.

Here is a decision table created with the Collect (Sum) hit policy:

The screenshot shows a decision table titled "Discount Percent". The table has three columns: a header row with "C+" and "Age" (highlighted in orange), a body row with "Enter Allowed Values", and four data rows. The data rows contain the following information:

C+	Age	Discount Percent
Enter Allowed Values		
1	<18	15
2	[18..45]	5
3	>45	10
4	>60	15

In this example, the last two rules match for an input age of 61. The decision table returns the sum of output values of these rules, that is, 25.

- **Multiple Hit Collect (Min)**

In a decision table with Collect (Min) hit policy, multiple rules with different output entries can match. The decision table returns the smallest output value among all matching rules.

Here is a decision table created with the Collect (Min) hit policy:

The screenshot shows a decision table titled "Discount Percent". The table has three columns: a header row with "C<" and "Age" (highlighted in orange), a body row with "Enter Allowed Values", and four data rows. The data rows contain the following information:

C<	Age	Discount Percent
Enter Allowed Values		
1	<18	15
2	[18..45]	5
3	>45	10
4	>60	15

In this example, the last two rules match for an input age of 61. The decision table returns the smallest output value among these rules, that is, 10.

- **Multiple Hit Collect (Max)**

In a decision table with Collect (Max) hit policy, multiple rules with different output entries can match. The decision table returns the largest output value among all matching rules.

Here is a decision table created with the Collect (Max) hit policy:

The screenshot shows a decision table interface with the following structure:

C>	Age	Discount Percent
1	<18	15
2	[18..45]	5
3	>45	10
4	>60	15

The table has a header row with columns for Condition (C>), Age, and Discount Percent. There are four rows of data. The first two rows (Age < 18 and [18..45]) both have a discount of 15, which is the maximum value among the matching rules.

In this example, the last two rules match for an input age of 61. The decision table returns the largest output value among these rules, that is, 15.

- **Multiple Hit Collect (Count)**

In a decision table with Collect (Count) hit policy, multiple rules with different output entries can match. The decision table returns the count of matching rules.

Here is a decision table created with the Collect (Count) hit policy:

C#	Age	Discount Percent
Enter Allowed Values		
1	<18	15
2	[18..45]	5
3	>45	10
4	>60	15

In this example, the last two rules match for an input age of 61. The decision table returns the count of matching rules, that is, 2.

Using Expressions

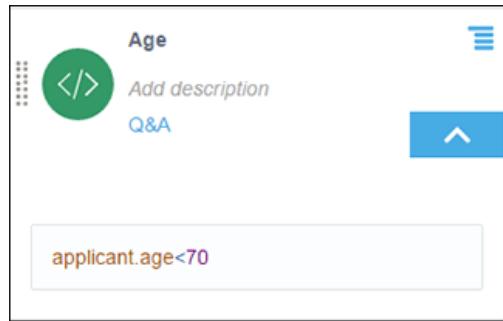
An expression is a logical notation, defined according to the syntax of FEEL, which evaluates to a single value. It may consist of one or more entities, such as a literal, constant, or variable, interconnected by zero or more operators. In Process, you can also use outputs of other decisions or built-in functions to define an expression.

To create a decision with the Expression notation:

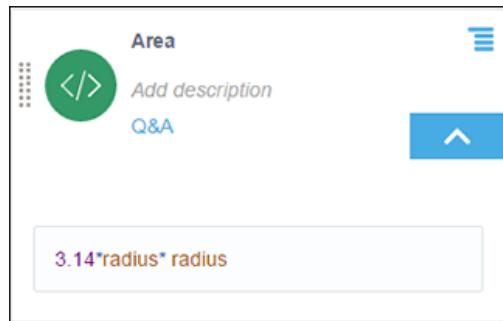
1. Click **Add new decision** in the Decisions bar and select **Expression** from the Create Decision window. See [Adding and Ordering Decisions](#).
2. In the Expression field, press **Ctrl+Space** to view an auto-suggest menu. You can use any decision outputs, variables, functions, and keywords listed in the menu to form your expression.
3. After the decision logic is complete, click **Save** to save and validate changes manually. Changes you make within the decision model are also automatically saved and validated from time to time. Errors, if any, are displayed within the decision.

Following are some examples of decisions created using simple expressions:

- If the *age* property of the variable *applicant* is less than 70, then the decision returns *true* else it returns *false*.



- The decision calculates the area using a constant and an input variable, *radius*.



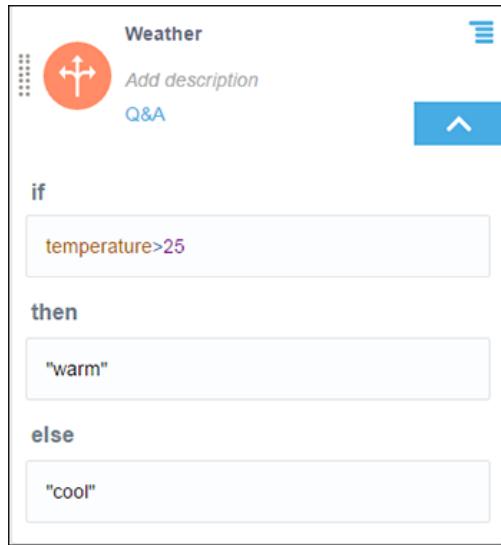
Using If-Then-Else Statements

An If-Then-Else expression is a logical notation that evaluates a test statement. It executes a primary expression if the test is *true* and a secondary expression if the test is not *true*.

To create a decision with the If-Then-Else logic:

- Click **Add new decision** in the Decisions bar and select **If-Then-Else** from the Create Decision window. See [Adding and Ordering Decisions](#).
- In the Expression field, press **Ctrl+Space** to view an auto-suggest menu. You can use any decision outputs, variables, functions, and keywords listed in the menu to define expressions in if, then, and else fields.
- Define expressions according to the FEEL syntax. See [Understanding FEEL](#).
- After the decision logic is complete, click **Save** to save and validate changes manually. Changes you make within the decision model are also automatically saved and validated from time to time. Errors, if any, are displayed within the decision.

In the following example, the input value of *temperature* determines the output of the If-Then-Else decision:



Using Functions

You can create functions to define specific operations that aren't available through built-in functions. In Process, decisions created using the Function notation return a value only when invoked from another decision.

To successfully invoke a function from another decision, the number and type of parameters in the function invocation must match those in the function definition.

The following example demonstrates a function implementation in Process. Here, the output decision invokes a function decision to calculate the seasonal discount percentage. The function decision contains the logic for regular discounts in the form of a decision table.

The screenshot shows the Oracle Decision Manager interface. At the top, there are two decision functions listed:

- Special Discount**: Described as "Special seasonal discount." It has an icon of a green circle with '</>' and a Q&A button.
- Discount**: Described as "Regular discount." It has an icon of a blue circle with 'fn' and a Q&A button.

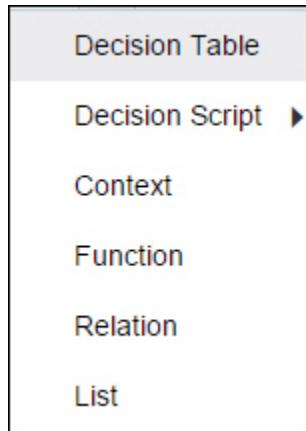
Below these, under the "Discount" function, there is a "Parameters" section with two entries: "c" (number type) and "p" (number type). There is also a "Body" section containing a "Decision Table".

A	c	p	Discount
1	-	>100	10
2	>=50	-	10
3	<50	<=100	5

The "Decision Table" header includes buttons for "New Row", "Delete Row", and "Enter Allowed Values".

To create a decision with the Function notation:

1. Click **Add new decision** in the Decisions bar and select **Function** from the Create Decision window. A function with empty Parameters and Body fields is created, with the Expression notation selected by default for the Body field.
2. In the Parameters field, click **Add** to add a new parameter. Enter a name for the parameter and select a data type for it. See [Data Types](#).
3. In the Body field, click **Change body** to change the logical notation. Select the required notation from available options.



4. Now, configure the logic for the selected notation. You can use input variables or built-in functions to define the logic.
5. Click **Save** to save and validate changes manually. Changes you make within the decision model are also automatically saved and validated from time to time. Errors, if any, are displayed within the decision.

Note:

Because a function decision by itself doesn't return a result, it's not an output decision. Therefore, you can't add function decisions to a decision service.

Using Contexts

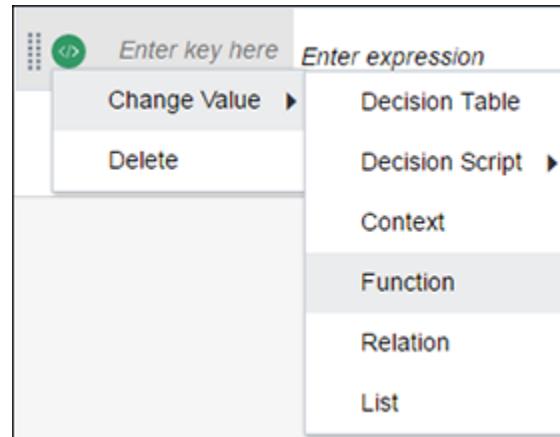
A **context** is a collection of one or more key-value pairs. Each pair is called a context entry. The key attribute within a context entry acts as an identifier to its corresponding value attribute.

You can use a context to collectively document all decision logic related to a particular scenario or entity. Say you need to document all relevant information related to a circle, such as diameter, circumference, and area within a decision. For this purpose, you can create a decision named *Circle* using the Context notation, and add expressions or logic to calculate diameter, circumference, and area as context entries.

In Process, a context is a logical notation with multiple outputs. It returns the outputs of all entries in a list format. By using the key attribute, you can also invoke a particular context entry from another decision.

To create a decision with the Context notation:

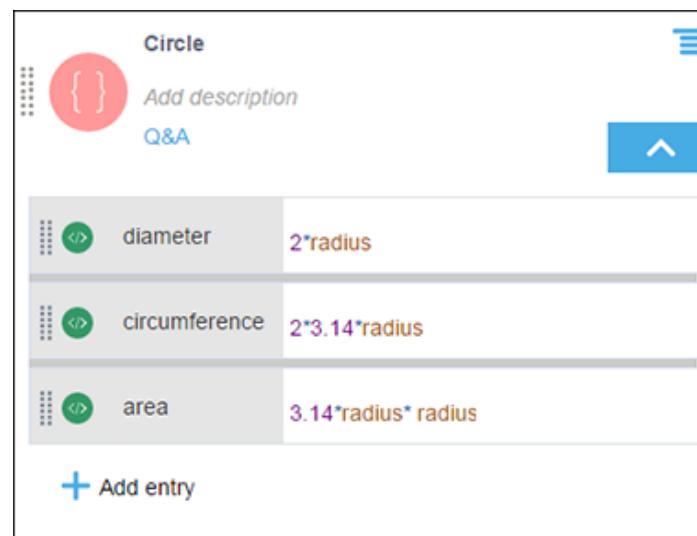
1. Click **Add new decision** in the Decisions bar and select **Context** from the Create Decision window. An empty Context is created.
2. Click **Add** to create a new context entry. A key-value pair is created with the expression notation selected by default in the Value field.
3. To change the logical notation for an entry, click the decision logic icon in the Key field to open the Change Value menu. Select a different notation from the available options.



4. In the Key field of a context entry, enter a unique name.
5. In the corresponding Value field, configure the logic for the selected notation. You can use input variables or built-in functions to define the logic.
6. Repeat steps 2 to 5 to add another entry into the context.
7. Drag and drop context entries to reorder them within the context.
8. Click **Save** to save and validate changes manually. Changes you make within the decision model are also automatically saved and validated from time to time. Errors, if any, are displayed within the decision.

To delete a context entry, click the decision logic icon in the Key field and select **Delete**.

The following image shows a context with multiple entries:



The output of this context is a list containing results of all three context entries. To reference a particular context entry (for example, area) from another decision, use *Circle.area*. Within a context, a value attribute of a context entry can't reference key attributes of entries below it; however, it may reference key attributes of context entries above it.

 **Note:**

If you add a function as one of the context entries, the context as a whole doesn't return a result. However, you will still be able to invoke results of other context entries throughout the decision model.

Using Lists

A **list** notation is a vertical list of elements, where each element is an independent logical notation. The output of a list notation contains outputs of all its elements. You can also invoke the output of a particular list element from another decision.

To create a decision with the list notation:

1. Click **Add new decision**  in the Decisions bar and select **List** from the Create Decision window. An empty list is created.
2. Click **Add**  to create a new list entry. An entry is created with the expression notation selected by default.
3. To change the logical notation for an entry, click **Edit**  for the particular entry. Select a different notation from the available options.
4. In the entry field, configure the logic for the selected notation. You can use input variables or built-in functions to define the logic.
5. Repeat steps 2 to 4 to add another entry into the list.
6. Click **Save** to save and validate changes manually. Changes you make within the decision model are also automatically saved and validated from time to time. Errors, if any, are displayed within the decision.

To delete a list entry, click **Edit** for the particular entry and select **Delete**.

The following example is a list of simple expressions containing prime numbers that are less than 10:

The screenshot shows a decision named "Prime Numbers". The description is "List of all prime numbers < 10.". Below the description, there is a "Q&A" section and a blue upward arrow button. The main area is titled "List" and contains four items: "2", "3", "5", and "7". At the bottom, there is a blue "Add item" button with a plus sign.

In a list of n elements, use $List_name[n]$ to invoke the n^{th} element from the beginning of the list, and use $List_name[-n]$ to invoke the n^{th} element from the end of the list. In this example, to invoke the list entry of 2, you can either use $\text{Prime Numbers}[1]$ or $\text{Prime Numbers}[-4]$.

You can also use suitable built-in list functions on a decision containing a List notation. For example, the following Expression decision returns the sum of all items in the *Prime Numbers* decision.

The screenshot shows a decision named "Sum". The description is "Sum of all prime numbers < 10.". Below the description, there is a "Q&A" section and a blue upward arrow button. The main area contains a single line of code: "sum(Prime Numbers)".

 Note:

- If you add a function as one of the list entries, then the list notation as a whole doesn't return a result. However, you will still be able to invoke results of other list entries throughout the decision model.
- According to the FEEL syntax, you can also define horizontal lists in expression fields across all notations. For example, a list of all prime numbers less than 10 can be defined as [2,3,5,7].

Using Relations

You can use a relation notation as convenient shorthand to represent multiple contexts.

A **relation** is a vertical list of similar contexts arranged horizontally. In other words, each row of a relation table is a context and each column consists of context entries, where the column name is the common key attribute for all cell entries under it that act as value attributes of respective contexts (rows). For details about contexts and key-value pairs, see [Using Contexts](#). In a relation, each cell entry is an independent logical notation.

In the output of a relation notation, outputs of all contexts within it are clearly distinguished. You can also invoke the output of a particular context or context entry from another decision.

To create a decision with the relation notation:

1. Click **Add new decision**  in the Decisions bar and select **Relation** from the Create Decision window. An empty relation table appears.
2. Use row/column controls available within the decision to add additional rows or columns. All cells have expression notation selected by default.
3. Enter a name for each column.
4. Select a cell and click **Change logic**  to change the logical notation for the cell. Select a different notation from the available options. Note that you cannot insert a decision table within a relation.
5. Within cells, configure the logic for selected notations. You can use input variables or built-in functions to define the logic.
6. Click **Save** to save and validate changes manually. Changes you make within the decision model are also automatically saved and validated from time to time. Errors, if any, are displayed within the decision.

The following relation contains the stock information for a particular brand of phone in the form of multiple contexts:

Relation		
Color	Count	Price
"Black"	2	500
"White"	3	700
"Rose Gold"	5	600

Similar to list notations, use either *Phone[1]* or *Phone[-3]* to access the entire context related to black-colored phones. To access all cell entries of a particular column, use the relation name in combination with the column name, for example, *Phone.Price* returns all entries of the Price column in a list. To access a particular context entry (for example, "Rose Gold"), use *Phone.Color[3]* or *Phone.Color[-1]*.

The following image is the result of the entire relation, which has results of each context listed separately:

Results	
Color	Black
Price	500
Count	2
Color	White
Price	700
Count	3
Color	Rose Gold
Price	600
Count	5

Note:

If you add a function as one of the cell entries, then the relation as a whole doesn't return a result. However, you will still be able to invoke results of other cell entries or contexts throughout the decision model.

Testing Decisions

After creating the decisions and supporting decisions within your decision model, you can verify that your decision model works the way you want by testing your model.

To test the decision model:

1. Click **Test** .
2. In the Test Decision Model pane, enter the input data to test your decision and click **Start Test**. The Decision Model Result pane displays decision results.
3. Click each green check mark to see the decision's result.
4. Click **Go Back**, and repeat steps 1 to 3 to test each decision rule's outcome.

Exposing Decisions as Services

To use your decision models in one or more Process applications, you must add at least one decision service in the decision model before you deploy the decision model. The decision service will expose one or more output decisions of your decision model as public REST APIs. A decision service consists of a set of input data and a set of output decisions from the containing decision model. You can use the decision services in your process to implement a decision model.

To configure a decision service:

1. In the decision model editor, expand the Services pane.
2. Click **Add**  to open the New Decision Service dialog box.
3. Enter a name for the decision service and click **Create**.
4. Drag and drop the output decision and input data that you want to expose through the service. You can add multiple values for the input data and the output decision.

To edit or delete an existing decision service, click **Options**  next to the service name. You can also view or copy the URL, request payload, and response payload of the decision service.

Deploying and Managing Decision Model Snapshots

Decision model snapshots are read-only copies of a decision model at a particular moment.

You can:

- Create a snapshot at any point while creating a decision model
- View the contents of a snapshot
- Delete a snapshot
- Export a snapshot to your local file system

Participants with space owner or space editor permissions can create a snapshot.

You can create and deploy a snapshot in any one of the following ways:

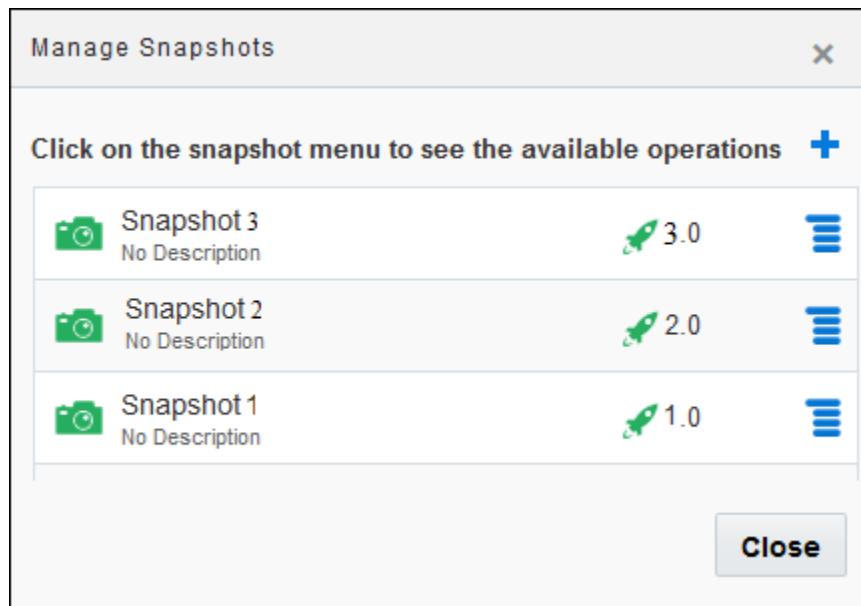
- From the Decision Model editor
- From the Manage Snapshots dialog

To create and deploy a snapshot from the Decision Model editor:

1. While creating a decision model, click **Deploy** to open the Deploy Application and Create Snapshot dialog.
2. Enter a name and description for the snapshot.
3. Assign a runtime version ID with which the snapshot will be deployed. A runtime version ID uniquely identifies a deployed snapshot among other deployed snapshots of the same model.
4. Optionally, select **Overwrite** to reuse an existing runtime version ID.
5. Click **Deploy** to create and deploy the snapshot.

To create a snapshot from the manage Snapshots dialog:

1. Click the Manage Snapshots icon  to open the Manage Snapshots dialog.



2. Click **Add**  to open the Create Snapshot dialog.
3. Enter a name and description for the snapshot and click **Create**.

To manage the existing snapshots:

1. Click the Manage Snapshots icon  to open the Manage Snapshots dialog box.
2. Right-click an existing snapshot. The available options are:

Option	Description
Deploy/Un-deploy	Deploy or un-deploy the snapshot.

Option	Description
View	View a read-only version of the decision model snapshot.
Delete	Delete the selected snapshot. You need to un-deploy a snapshot that is already deployed before deleting it. Warning: If you delete a snapshot that is currently in use in one or more applications, the reference to the associated decision model does not get deleted and remains in the Process application. This may result in errors during runtime when the application tries to call the decision service associated with a decision model snapshot that has previously been deleted from the DMN server.
Export	Export any snapshot of the decision model, deployed or not, to your local file system.

 **Note:**

You can import a previously exported snapshot as a new decision model and edit its content.

Adding Decisions to Applications and Processes

After creating a decision model, you can use it in a process within your application.

To add a decision model in a process:

1. On the Process Editor's elements palette, expand the System elements.
2. Drag and drop a Decision flow element into the process.
3. Implement the Decision element:
 - a. Click the element and select **Properties, General**.
 - b. Select a decision model from the list, or click **Add +** next to the Decision Model field to display a list of deployed decision models.
 - c. Select the decision from the list in the Use a Decision Model dialog box, enter a name for the decision model, and click **Use**.
 - d. Select a decision service to use from the list of services defined for the selected decision model. Process creates all the metadata definitions such as importing the required types and defining the connector to the service.
4. Click **Save**  to save your changes.

Importing and Exporting Decision Models

You can import and export decision models as DMN files. With this feature, you can share decision models with other users directly through the file system.

Topics

- [Importing a Decision Model](#)
- [Exporting a Decision Model](#)

Importing a Decision Model

You can import a decision model that was previously exported and saved as a DMN file.

To import a decision model:

1. On the Composer Home page, click **Create**, select **Import**, and then select **Decision Model**.
2. Click **Browse**, then select the decision model file (.DMN) you want to import.
3. Enter a name for this decision model.
4. Select the space where you want to store your imported decision model.
5. Click **Import**.

Exporting a Decision Model

Exporting a decision model to your local file system enables you to share decision models.

To export a decision model to a DMN file:

1. On the Composer Home page, find the decision model you want to export.
2. Click **Options**, then select **Download**.
3. Select **Save File**, choose a location on your local file system, and click **Save**.

11

Creating Business Rules

Business rules determine dynamic decisions that enable you to automate policies, computations, and reasoning. These business rules can be individual if/then rules or spreadsheet-like decision tables.

Note:

Process provides two editors related to decisions: the **Decision Model editor** (recommended) described in [Creating Decisions](#), and the **Oracle Business Rules editor** described below.

Topics

- [About Business Rules](#)
- [What You Can Do Using the Business Rules Editor](#)
- [Typical Workflow for Creating Business Rules](#)
- [Creating a Decision in Business Rules](#)
- [Creating a Value Set](#)
- [Creating a Range Value Set](#)
- [Creating Global Variables](#)
- [Creating an If/Then Rule in Business Rules](#)
- [Creating a Decision Table in Business Rules](#)
- [Assigning a Business Rule to a Process Component](#)
- [Typical Workflow for Using Advanced Business Rule Features](#)
- [Using Advanced Conditions](#)
- [Using Advanced Actions](#)
- [Editing Rule and Decision Table Properties](#)
- [Adding Conflict Resolutions to Decision Tables](#)
- [Editing Decision Properties in Business Rules](#)

About Business Rules

Business rules enable you to automate policies, computations, reasoning, and task assignment. Each rule has conditions (*if* statements) and actions (*then* statements). Rules can be individual if/then statements or organized into spreadsheet-like decision tables, in which rows represent conditions and actions, and columns match condition values to action alternatives.

For example, if/then rules for a loan approval application might be:

```

IF income < 20,000 THEN create outcome: Reject
IF income between 20,000 and 50,000 and creditGood is true THEN create outcome:
Manual
IF income between 20,000 and 50,000 and creditGood is false THEN create outcome:
Reject
IF income between 50,000 and 100,000 and creditGood is true THEN create outcome:
Accept
IF income between 50,000 and 100,000 and creditGood is false THEN create outcome:
Manual
IF income >=100,000 THEN create outcome: Accept

```

A decision table that includes the same if/then rules might look like this:

	R1 ▾	R2 ▾	R3 ▾	R4 ▾	R5 ▾	R6 ▾
Conditions						
Decision1.in.income ▾	<20,000	[20,000..50,000)		[50,000..100,000)		>=100,000
Decision1.in.creditGood ▾	-	true	false	true	false	-
Actions						
create Decision1.out ▾	✓	✓	✓	✓	✓	✓
outcome	"Reject"	"Manual"	"Reject"	"Accept"	"Manual"	"Accept"

A square bracket includes the range endpoint. A parenthesis excludes the range endpoint. A dash means that the value in that cell doesn't matter.

Note:

If all you need is one if/then rule with one condition and one action, you can use a [gateway](#) in your process instead. For examples of how gateways work independently of rules, see [About Business Processes](#). After you create business rules, you use a gateway to take different paths in your process based on the outcomes. See [Assigning a Decision to a Process Component](#).

Understanding business rules involves understanding the following concepts.

Rules Concept	Description
Decision	A decision is a container for if/then rules and decision tables that use the same input and output data objects. A decision exposes these data objects as a reusable service that multiple business processes can invoke.
Input and Output	When you create a decision , you must specify the input and output data objects. For example, the if/then rule and decision table examples above have the input data objects <code>income</code> and <code>creditGood</code> and the output data object <code>outcome</code> . You can create new data objects or use preexisting data objects.
Condition	The <i>if</i> statements of an if/then rule or decision table are called conditions . Conditions usually operate on input data objects.
Action	The <i>then</i> statements of an if/then rule or decision table are called actions .

Rules Concept	Description
Value Set	A value set defines a list of allowed values or ranges for a data object. For example, the value set for income includes the ranges less than 20,000, between 20,000 and 50,000, between 50,000 and 100,000, and more than 100,000. The value set for outcome includes the values Reject, Manual, and Accept.
Global	Use global variables to share information among several if/then rules and functions. For example, if a 10% discount for Gold customers is used in several if/then rules, you can create a Gold Discount global variable with the value 10.

What You Can Do Using the Business Rules Editor

The business rules editor allows you to easily create and edit if/then rules and decision tables.

You can't access the business rules editor until you create a new decision on the Application Home tab. See [Creating a Decision](#).

The rules editor opens as a tab and therefore, the main toolbar is visible. This toolbar contains actions that apply to the business process application as a whole and are available in addition to the actions available in the rules editor.

The rules editor is divided into two areas: *component bar* and *canvas*.

About the Component Bar

The rules editor provides quick access to components related to rules design.



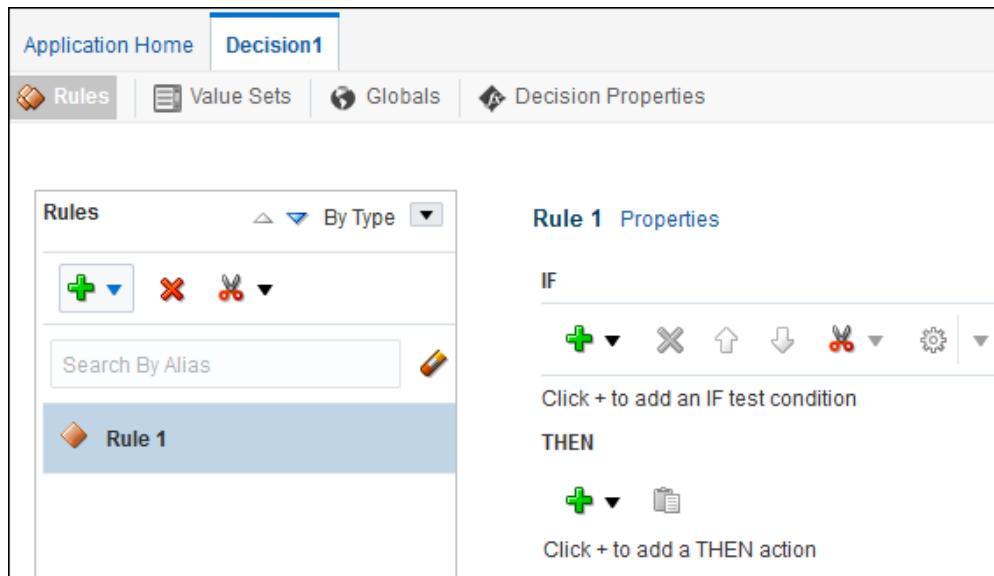
Components include:

Component	Description
Rules	Create if/then rules and decision tables .
Value Sets	Create value sets and range value sets .
Globals	Create global variables .
Decision Properties	Edit decision properties .

About the Rules Editor Canvas

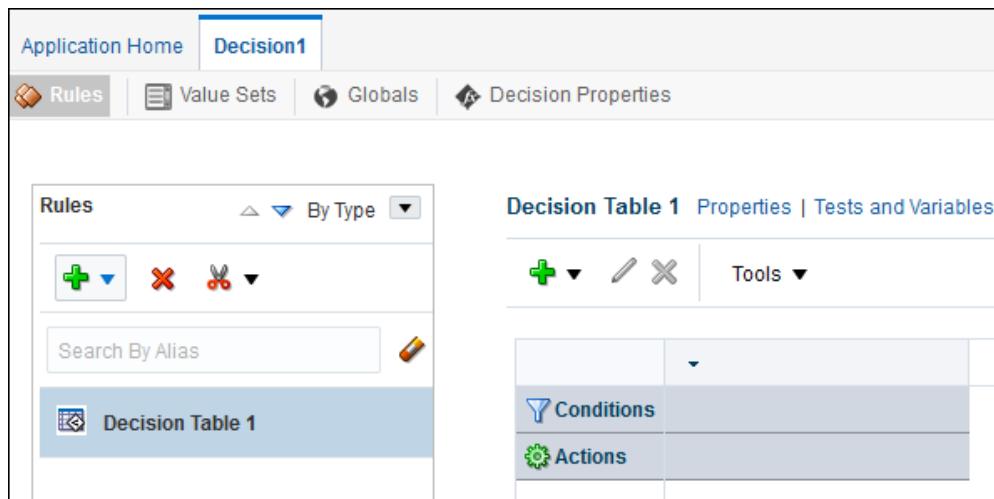
The canvas is the central area of the rules editor. The contents of the canvas change as you click different items in the component bars.

What you see most often is the Rules canvas. When you create a new if/then rule, the Rules canvas looks like this:



See [Creating an If/Then Rule](#).

When you create a new decision table, the Rules canvas looks like this:



See [Creating a Decision Table](#).

Typical Workflow for Creating Business Rules

Use this typical workflow as your guide when creating business rules.

Task	Description
Create a decision	A decision is a container for if/then rules and decision tables that use the same input and output data objects. You create a new decision from the Application Home tab.

Task	Description
Create value sets and range value sets	A value set defines a list of allowed values or ranges for the input and output data objects of a decision.
Create global variables	A global variable contains information shared among several if/then rules and functions.
Create if/then rules	If/then rules perform one or more <i>then</i> actions based on one or more <i>if</i> conditions.
Create a decision table	If/then rules can be organized into spreadsheet-like decision tables, in which rows represent conditions and actions, and columns match condition values to action alternatives.
Add the decision to a process	You specify the decision in a Decision process component.

For advanced features, see [Typical Workflow for Using Advanced Business Rule Features](#).

Creating a Decision in Business Rules

You create a decision by specifying the input and output data objects upon which the if/then rules and decision tables within the decision act.

You must plan what the input and output data objects of a decision are going to be before you create the decision. Once a decision is created, you can't add or delete input and output data objects.

To create a new decision:

1. Go to the Application Home tab and make sure you're in **Edit** mode if you're editing a shared application.
2. Click **Decisions**, click **Add** , and then **Create Rule**.
3. Enter a name for the rule.

The field is populated with a default name, such as *Rule*, *Rule1*, and so on. You can use this name or change it.

4. For each input and output data object:
 - a. Click **Add Data Object** to open the New Data Object dialog box.
 - b. Enter a name for the data object that represents the data object.
 - c. Select **Input** or **Output** from the drop-down list.
 - d. Select the data object type from the **Type** drop-down list: **int** for whole numbers, **long** for large whole numbers, **float** for large decimal numbers, **double** for decimal numbers, **boolean** for true or false, **short** for small whole numbers, **string** for text, or **object** for an existing business object.

Forms have three generic numeric fields: Money, Quantity, and Number. Money and Number are of type double, and Quantity is of type int. When you perform data association in your process, you can map any numeric type to float, but you can only map int to int. Therefore, you should select **float** for a numeric data object that comes from a form unless you're certain that the form uses a Quantity field.

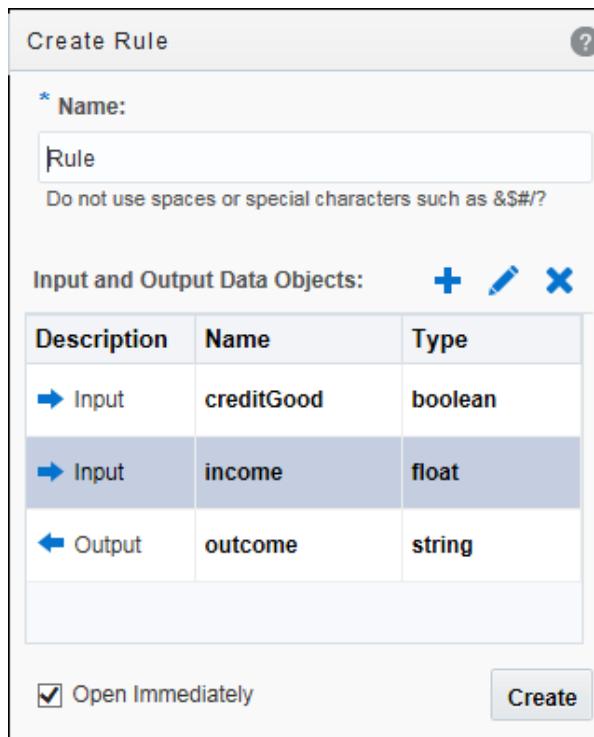
The **object** option appears only if you have created a business object. A schema (XSD) file that defines a business object for use in a decision must define a complex type and must contain exactly one element of that type. However, the same business object cannot be used multiple times as an input object to a rule.

Want to learn more about data objects and business objects? See [Managing Application Data](#).

e. Click **Add**.

A rule must have at least one input data object and one output data object.

Here is an example of the Create Rule dialog box with three data objects added:



5. Deselect the **Open Immediately** check box to go to the Application Home tab instead of the Business Rules Editor and click **OK**.

The Create Rule dialog box closes. If **Open Immediately** was selected, the new rule opens in the Business Rules Editor.

6. If **Open Immediately** was deselected, click the name of the rule to open it in the Business Rules Editor.

To view information about the input and output data objects later, see [Editing Decision Properties in Business Rules](#).

Creating a Value Set

A **value set** defines a list of allowed values for the input and output data objects of a decision.

To create a new value set:

1. In the Business Rules Editor, click **Value Sets**.

2. Select **Value Set** from the Add Value Set drop-down list.
3. Change the name from the Value Set *n* default. This is optional.
4. Enter a description. This is optional.
5. Select a data type from the **Data Type** drop-down list:
 - **String** — text
 - **int** — whole numbers
 - **double** — decimal numbers
 - **boolean** — true or false
 - **char** — single characters
 - **byte** — eight bits, the smallest unit of data
 - **short** — small whole numbers
 - **long** — large whole numbers
 - **float** — large decimal numbers
 - **Date** — dates only
 - **Time** — times only
 - **DateTime** — dates and times

The data type must match the data type of the data object for which you're creating the value set.

6. Check **Include Disallowed Values in Tests** if the rule or decision table using the value set performs an action based on a disallowed value.
7. For each value:
 - a. Click **Add Value**. A new row is added to the Values table.
 - b. Enter the value.
 - c. Enter an alias for the value. By default the alias is the same as the value.
For example, for the value *Manual*, you could add this alias: Loan officer evaluates the application.
 - d. Deselect the **Allowed in Actions** box if the value won't be associated with an action.
 - e. Add a description. Every value set has an *otherwise* value, which you cannot delete.

Here is an example of a String value set with the Description column omitted:

	Value	Alias	Allowed in Actions
	otherwise	otherwise	<input checked="" type="checkbox"/>
	"Accept"	Accept	<input checked="" type="checkbox"/>
	"Reject"	Reject	<input checked="" type="checkbox"/>
	"Manual"	Loan officer evaluates the application.	<input checked="" type="checkbox"/>

8. When you're done adding values, click **Save and Publish**.

 **Note:**

To delete a value, select the row you want to delete. The **Delete**  icon gets activated only after you select the row.

To learn how to define a value set where each value in the set is a range of values, see [Creating a Range Value Set](#).

Creating a Range Value Set

A **range value set** defines a list of allowed value ranges for the input and output data objects of a decision.

To create a new range value set:

1. In the Business Rules Editor, click **Value Sets**.
2. Select **Range Value Set** from the **Add Value Set** drop-down list.
3. Optional: Change the name from the Value Set *n* default.
4. Optional: Enter a description.
5. Select a data type from the **Data Type** drop-down list:
 - **String** — text
 - **int** — whole numbers
 - **double** — decimal numbers
 - **char** — single characters
 - **byte** — eight bits, the smallest unit of data
 - **short** — small whole numbers
 - **long** — large whole numbers
 - **float** — large decimal numbers
 - **Date** — dates only
 - **Time** — times only
 - **DateTime** — dates and times

The data type must match the data type of the data object for which you're creating the value set.

6. Check **Include Disallowed Values in Tests** if the rule or decision table using the value set performs an action based on a disallowed value.
7. To define the ranges:
 - a. Click **Add Value**.
A new row is added to the Range Values table.
 - b. Enter the range endpoint.
 - c. Select or deselect the **Included Endpoint** box.

For example, to define a range of less than but not including 50,000, enter 50,000 as the range endpoint and deselect the **Included Endpoint** box.

- d. Deselect the **Allowed in Actions** box if the range won't be associated with an action.
- e. Optional: Enter an alias for the range.
By default the alias is the same as the range.
- f. Optional: Add a description.

Every range value set has an endpoint **-Infinity**, which you can't delete.

Here is an example of a range value set with the Description column omitted:

	End Point	Included Endpoint	Allowed in Actions	Range	Alias
	100,000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	$\geq 100,000$	$\geq 100,000$
	50,000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[50,000..100,000)	[50,000..100,000)
	20,000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[20,000..50,000)	[20,000..50,000)
	-Infinity	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	$< 20,000$	$< 20,000$

A square bracket includes the range endpoint. A parenthesis excludes the range endpoint.

8. When you're done adding values, click **Save and Publish**.

To learn how to define a value set where each value in the set is a single value, see [Creating a Value Set](#).

Creating Global Variables

Use global variables, also called globals, to share information among several rules and functions. For example, if a 10% discount for Gold customers is used in several rules, you can create a Gold Discount global with the value 10.

To create a new global variable:

1. In the Business Rules Editor, click **Globals**.
2. Click **Add Global**.
3. Enter a name for the global.
4. Optional: Enter a description.
5. Enter or select a value:
 - Enter a value in the **Value** field.
 - Select from the **Value Set** drop-down list, then select from the **Value** drop-down list.
6. If you entered a value in the previous step, select the data type from the **Type** drop-down list:
 - **decision-name.in** — input data objects of a decision

- **decision-name.out** — output data objects of a decision
- **boolean** — true or false
- **byte** — eight bits, the smallest unit of data
- **char** — single characters
- **double** — decimal numbers
- **float** — large decimal numbers
- **int** — whole numbers
- **long** — large whole numbers
- **short** — small whole numbers
- **String** — text
- **Boolean** — Boolean Java type
- **BigDecimal** — BigDecimal Java type
- **BigInteger** — BigInteger Java type
- **Double** — Double Java type
- **Float** — Float Java type
- **Int** — Int Java type
- **Long** — Long Java type
- **Short** — Short Java type

Selecting a data type limits the choices in the **Value Set** drop-down list to value sets of that type.

7. Deselect the **Final** box to permit the global to be modified in an if/then rule or decision table action.
8. Check the **Constant** box to specify that the value is treated literally in expressions. Value set values and range endpoints are always constant.
9. Repeat all the previous steps for each global variable you want to add.
10. When you're done adding global variables, click **Save and Publish**.

In addition to creating global variables, be sure to check out how to [create a value set](#), [create a range value set](#), and use global variables in [advanced conditions](#).

Creating an If/Then Rule in Business Rules

An if/then rule has one or more conditions (if statements) that together are true or false, and one or more actions (then statements) that are applied if the conditions are true.

To create a new if/then rule:

1. In the Business Rules Editor, click **Rules**.
2. Click **General If/Then Rule** in the canvas or select **General Rule** from the **Add** drop-down list in the left pane.

The IF toolbar, condition area, THEN toolbar, and action area appear. The default name is Rule 1.

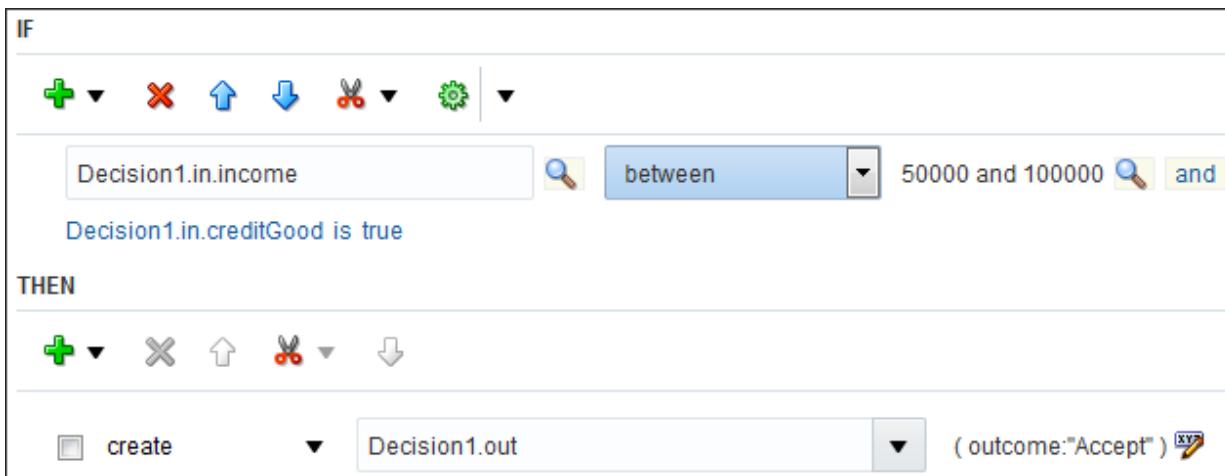
3. Optional: To change the if/then rule name, click **Properties** next to the name, edit the name, and click **OK**.
4. For each condition you want to add:
 - a. In the IF toolbar, select the **simple test** condition from the **Add** drop-down list.
For descriptions of other conditions, see [Using Advanced Conditions](#).
Two fields and an operator drop-down list appear.
 - b. Select an operator.
Basic numeric operators are **is**, **isn't**, **more than**, **same or more than**, **less than**, **same or less than**, and **between**. You can also use **between** for dates and times. Basic text operators are **ends with**, **equals ignore case**, **matches**, and **starts with**. The **matches** operator specifies that the text string on the left includes the text string on the right.
For descriptions of the **in** and **contains** operators, see [Using Advanced Conditions](#).
 - c. Click **Left Value** or **Right Value** to browse for value sets, globals, functions, and decision data objects.
The **between** operator prompts for two operands when you click **Right Value**.
You can find the input data objects you specified in [Creating a Decision](#) by expanding **decision-name.in** in the Condition Browser.
- When you add multiple conditions, the **and** operator appears between them by default.
5. Click **and** to change it to **or** if necessary.
6. Use **Move Up** and **Move Down** to change the position of the selected condition.
Conditions and the **and** and **or** operators between them are applied in order. For example, A and B or C and D is applied as (((A and B) or C) and D). For other grouping and nesting options, see [Using Advanced Conditions](#).
7. In the THEN toolbar, select an action from the **Add** drop-down list:
 - **create** — Creates a new data object and assigns a value to it. Use this action for an output data object, unless it's also an input data object.
 - **assign** — Assigns a value to an existing global or local variable.
 - **call** — Calls a function.
 - **modify** — Modifies the value of an existing data object.
For descriptions of other actions, see [Using Advanced Actions](#).
An output data object can only be created once. Therefore, if multiple rules and decision tables affect the same output data object, use **create** in an initial action, then use **modify** in the rules and decision tables. See [Editing Decision Properties](#) for details about initial actions.
8. Select a data object from the **Select a Target** drop-down list to the right of the action.
The output data objects you specified in [Creating a Decision](#) are in **decision-name.out**.
9. Click **Edit Properties**, click **Value** to browse for value sets, then click **OK**.
10. Repeat the previous three steps to add additional actions if necessary.

11. Use **Move Up** and **Move Down** to change the position of the selected action.

All actions are applied, in order.

12. When you're done creating if/then rules, click **Save** and **Publish**.

Here is an example of an if/then rule with two conditions:



You can [create a decision table](#) to organize your if/then rules into a spreadsheet-like format, in which rows represent conditions and actions, and columns match condition values to action alternatives. You can also [modify the properties of a decision table](#).

Creating a Decision Table in Business Rules

A decision table organizes if/then rules into a spreadsheet-like format, in which rows represent conditions and actions, and columns match condition values to action alternatives.

To create a new decision table:

1. In the Business Rules Editor, click **Rules**.
2. Click **Decision Table** in the canvas or select **Decision Table** from the **Add** drop-down list in the left pane.

An empty decision table appears. The default name is Decision Table 1.

3. Optional: To change the decision table name, click **Properties** next to the name, edit the name, and click **OK**.
4. For each condition you want to add:

- a. In the decision table toolbar, select **Add Condition** from the **Add** drop-down list.

The Add/Modify Condition dialog box opens.

- b. Type a condition value in the **Condition** field or expand the folders in the Name column to browse for functions and decision data objects.

To find the input data objects you already specified (see [Creating a Decision](#)), expand **decision-name.in**.

c. Click **Select or Edit Value Set**.

If the data object for this condition is boolean (true or false), you can skip this step and the next.

The value set page of the Add/Modify Condition dialog box opens.

d. Select an existing value set from the list on the left, or select **Local Value Set** or **Local Range Value Set** to create a new value set that applies only to this condition.

See [Creating a Value Set](#) or [Creating a Range Value Set](#).

e. Click **Done**.

The Add/Modify Condition dialog box closes.

f. If this is the first condition, select **Add Rule** from the **Add** drop-down list repeatedly, until you see an **Rn** column for each value or value combination you need.

For subsequent conditions, you might need to add additional columns to account for more condition combinations.

g. Click each table cell and select one or more values from the drop-down list that appears.

If the value in a cell doesn't matter, select **All** from the list. A dash appears in the cell.

5. To add conditions that apply to all rules or to create variables to make rules more readable, expand **Tests and Variables**.

For example, if all rules in your decision table apply only to legal adults, you can add a condition such as `Duration.years between(Person.birthdate, CurrentDate.date) same or more than 18`.

See [Using Advanced Conditions](#).

6. In the decision table toolbar, select one of the **Add Action** commands from the **Add** drop-down list:

- **create** — Creates a new data object and assigns a value to it. Use this action for an output data object, unless it's also an input data object.
- **call function** — Calls a function.
- **modify** — Modifies the value of an existing data object.

For descriptions of other actions, see [Using Advanced Actions](#).

An output data object can only be created once. Therefore, if multiple rules and decision tables affect the same output data object, use **create** in an initial action, then use **modify** in the rules and decision tables. Want to learn more about initial actions? See [Editing Decision Properties](#).

The Action Editor dialog box opens.

7. Select a function or decision from the data objects list.

You can find the output data objects you specified in [Creating a Decision](#) by selecting **decision-name.out**.

8. To use different action values for different conditions, select the **Parameterized** check box for each function or decision argument.

9. Click **OK**.

10. For each action cell:

- Click the cell and click **Select Value**.

The Condition Browser appears.

- Expand the desired value set and select a value.

- Click **OK**.

You can't create a local value set for an action. You must use an existing value set.

11. Add more actions if necessary.

All actions are applied, in order.

12. If conflicts occur, see [Adding Conflict Resolutions to Decision Tables](#).

13. Use these commands on the **Tools** menu to clean up or rearrange the decision table:

- Identify Possible Missing Rules:** Checks for condition combinations you might have missed.
- Order Rules by Value:** Orders condition combinations by value from left to right and merges cells with the same value where possible.
- Generate Unique Rules:** Displays all condition value combinations by eliminating merged and all-value (dash) cells.
- Switch Rows to Columns:** Pivots the table so that rules are rows and conditions and actions are columns.

14. When you're done creating the decision table, click **Save** and **Publish**.

Here's an example of a decision table with two conditions:

	R1 ▾	R2 ▾	R3 ▾	R4 ▾	R5 ▾	R6 ▾
Conditions						
Decision1.in.income ▾	<20,000	[20,000..50,000)		[50,000..100,000)		≥100,000
Decision1.in.creditGood ▾	-	true	false	true	false	-
Actions						
create Decision1.out ▾	✓	✓	✓	✓	✓	✓
outcome	"Reject"	"Manual"	"Reject"	"Accept"	"Manual"	"Accept"

Assigning a Business Rule to a Process Component

For business rules to run, these rules must be part of a business process. In most cases, an exclusive gateway handles the different outcomes of a decision in business rules.

To assign a business rule to a process component:

- On the process editor's Elements Palette, expand the System elements.
- Drag and drop a Rule element into the process.

3. Implement the Rule element:
 - a. Click the element and select **Properties, Implementation**.
 - b. Click **Browse** next to the Business Rule field to display a list of rules.
 - c. Select the rule from the list in the Business Rule Browser and click **OK**.
4. Click **Gateway** to expand the list of available gateways.
5. Drag and drop an **Exclusive Gateway** element into the process immediately after the Rule in the flow.
6. Create a gateway path for each rule outcome by dragging the arrow icon that appears when you select the gateway.

Make sure the default path (with the slash) connects to the correct flow element.
7. Create a data object for the gateway:
 - a. Select the gateway and click **Data Objects** to open the Data Objects dialog.
 - b. Click **Add** to open the New Process Data Object dialog.
 - c. Type a name, beginning with a lowercase letter.

For example, type `ruleOutcome`.
 - d. Select the type that matches the type of the output data object in the rule.

In most cases, the type is **boolean** (true or false) or **string** (text).
 - e. Click **OK**, then click **Close**.
8. Implement each non-default gateway path:

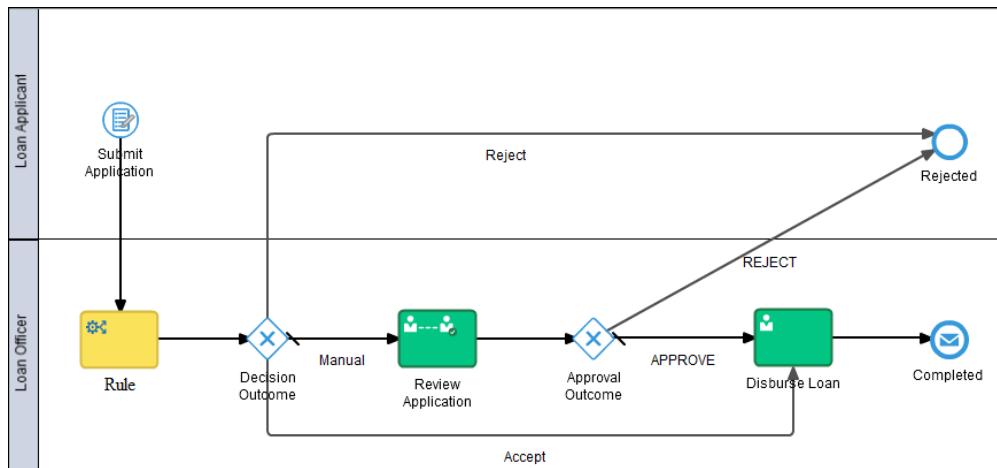
You don't need to implement the default gateway path, but you should name it.

 - a. Click the path, click **Edit** on the path, and then click **Properties, Implementation**.
 - b. Type a name for the path.
 - c. Click **Edit** next to the Condition field to open the Expression Editor.
 - d. Type an expression that tests whether the gateway data object has one of the rule outcome values.

For example, type `ruleOutcome == "Accept"`. Note the double equal sign.
 - e. Click **OK**.
 - f. Close the implementation pane.
9. Associate the rule output data with the gateway data object:
 - a. Click the Rule and click **Data Association**.
 - b. Expand the Process and Data Objects nodes in the right pane.
 - c. Drag and drop the gateway data object into the Outputs field.

You can associate Rule inputs as well, which often come from a form. If the form is referenced in the start event or another process component, the form data object appears in the right pane. Drag and drop each form field from the right pane to the corresponding Inputs field.
 - d. Click **Apply**.
10. Click **Save and Publish**.

[Creating a Decision Table](#)



Typical Workflow for Using Advanced Business Rule Features

Use this typical workflow as your guide for using the advanced features of business rules.

Task	Description
Use advanced conditions	You can group multiple conditions and create variables to make rules more readable. Use advanced conditions in if/then rules and in the Tests and Variables section in decision tables.
Use advanced actions	Advanced actions let you do more with rules than create or modify output data. For example, you can call built-in functions, assign values to variables, and perform functions in an expression.
Edit rule and decision table properties	You can: <ul style="list-style-type: none"> Change the name Add a description Set the priority Enable or disable a rule or decision table Specify the decision table conflict resolution policy Allow or disallow condition combination gaps in decision tables Set an effective date range
Resolve decision table conflicts	When condition combinations for two or more rules have overlapping values and different actions, you can: <ul style="list-style-type: none"> Select which conflicting rules override or run before the others Automatically override general cases with special cases Ignore the conflicts
Edit decision properties	You can view input and output data object properties and add actions that apply to the entire decision.

For basic features, see [Typical Workflow for Creating Business Rules](#).

Using Advanced Conditions

Advanced conditions let you group multiple conditions and create variables to make rules more readable. Use them in if/then rules and in the **Tests and Variables** section in decision tables.

Before you use advanced conditions, make sure you understand how to use simple tests with basic operators. See [Creating an If/Then Rule](#).

A variable created in a condition applies only to the if/then rule or decision table that includes the condition. However, you can create global variables, which are available to any if/then rule or decision table in a decision. See [Creating Global Variables](#).

Conditions can be divided into four groups:

- Simple
- Nested
- Where-Clause
- Variable

Simple Conditions

Simple conditions involve no nesting and create no variables.

Condition	Description	Example
simple test	<p>Uses an operator to compare two or more values.</p> <p>For basic operators, see Creating an If/Then Rule.</p> <p>Advanced operators involve lists:</p> <ul style="list-style-type: none"> • in — Tests whether a value is in a list. The left value is a data object or business object with one value. The right values that comprise the list are literal. • contains — Tests whether a list contains a value. The left value is a data object or business object that has multiple values. The right value can be a data object or business object with one value or a literal value. 	Employee.title in Manager, Director, Executive PurchaseOrder.items contains Product.itemnumber
boolean expression	Evaluates the expression to true or false. The simplest example is the name of a boolean data object or value set.	Person.isUsCitizen Duration.years between(Person.birthdate, CurrentDate.date) >= 18

Nested Conditions

Nested conditions group multiple inner conditions within an outer condition.

To create a nested condition:

1. Click **Add** and select one of the nested conditions from the list.
2. Click **Add** within the outer condition and select a condition from the list.
 - To add inner conditions after the first, select one of the inner conditions and click the main **Add** icon.
 - To add a condition at the same level as the outer condition, select the outer condition and click the main **Add** icon.

Condition	Description	Example
nested test	Adds the outer condition the following test is true . The inner conditions are joined by and by default. You can click and to change it to or .	the following test is true Person.isUsCitizen and Duration.years between(Person.birthdate, CurrentDate.date) same or more than 18
negated test	Adds the outer condition the following test isn't true . The inner conditions are joined by and by default. You can click and to change it to or .	the following test isn't true Product.inventory is 0 or Product.discontinued
all of the following	Adds the outer condition all of the following are true . The inner conditions are joined by and logic, which you can't change.	all of the following are true Person.isUsCitizen Duration.years between(Person.birthdate, CurrentDate.date) same or more than 18
any of the following	Adds the outer condition any of the following are true . The inner conditions are joined by or logic, which you can't change. At least one inner condition must be true.	any of the following are true Person.hasCollegeDegree Person.yearsJobExp same or more than 5

Where-Clause Conditions

Like nested conditions, where-clause conditions group multiple inner conditions within an outer condition. The where clause in the outer condition iterates over a specific class of business objects or over all business objects that are accessible in the decision.

Condition	Description	Example
there is an object where	Adds the outer condition there is a business object where . The default business object is decision-name.in . You can change the business object to any object or data object that's accessible in the decision. The inner conditions are joined by and by default. You can click and to change it to or .	there is a PurchaseOrder.items where Product.price more than 250 or Product.quantity more than 100

Condition	Description	Example
there is no object where	Adds the outer condition there is no business object where . The default business object is decision-name.in . You can change the business object to any object or data object that's accessible in the decision. The inner conditions are joined by and by default. You can click and to change it to or .	there is no PurchaseOrder.items where Product.inventory is 0 or Product.discontinued
there is a case where	Adds the outer condition there is a case where , which iterates over all business objects that are accessible in the decision. The inner conditions are joined by and by default. You can click and to change it to or .	there is a case where Product.sales more than 1,000,000 and Product.discontinued
there is no case where	Adds the outer condition there is no case where , which iterates over all business objects that are accessible in the decision. The inner conditions are joined by and by default. You can click and to change it to or .	there is no case where Product.sales less than 1,000 and Product.discontinued is false

Variable Conditions

Variable conditions don't perform tests. Variable conditions create convenient variables that other conditions can use.

Condition	Description	Example
variable	Creates a variable that represents a business object or the result of a test.	Good Credit = Decision1.in.CreditGood Bad Credit = Decision1.in.CreditGood is false
is a	Creates a variable that represents a business object for the purpose of comparing it to another business object of the same type. For example, to find the region with the highest sales, you must compare each region with all the others. To compare two different regions, you need two region variables.	IF region1 is a RegionalSales and there is no case where region2 is a RegionalSales and region2.amount > region1.amount THEN modify region1.highestAmount = true

Condition	Description	Example
aggregation	Creates a variable that represents the aggregation of a set of values with an optional where clause. The aggregation function can be sum , average , minimum , maximum , collection , or count . The collection function creates a list that the variable represents.	SalesTotalUSA is the sum of RegionalSales.amount where RegionalSales.country is "USA"

Using Advanced Actions

Advanced actions let you do more with rules than create or modify output data.

Before you use advanced actions, make sure you understand how to use basic actions as described in [Creating an If/Then Rule](#).

Action	Description	Example
create	<p>Creates a new data object and assigns a value to it. Use this action for a data object created by the rule.</p> <p>The data object can be an output of the decision or an input to another rule. You can specify some or all of the data object's values.</p> <p>A validation error occurs if an output data object isn't created, unless it's also an input data object.</p>	<pre>create Decision1.out (outcome:"Accept")</pre>
call	<p>Calls a built-in function for side-effects only. You can't access the return value of the function, if any.</p>	<pre>call print("Hello world!")</pre>
modify	<p>Modifies the value of an existing data object.</p> <p>The data object can be an output of the decision or an input to another rule. You can specify some or all of the data object's values.</p> <p>Changed values are returned for an output data object. Rules may run if they test the changed values. Be careful changing values in the same rule that tests those values. This can result in rules running repeatedly.</p>	<pre>modify Decision1.in (creditGood:true)</pre>

Action	Description	Example
assign	<p>Assigns a value to an existing global or local variable.</p> <p>You can't use assign to change the value of a variable defined in a condition. See Using Advanced Conditions.</p> <p>If you're assigning a value to a global variable, make sure Final is deselected in its definition. See Creating Global Variables.</p> <p>Assigning values to data objects isn't recommended. Use modify instead.</p>	assign GlobalDiscount = 15
remove	<p>Deletes an input data object.</p> <p>The deleted data object can't be matched by other rules, except for a <i>there is no object where</i> test.</p>	remove Decision1.in
assert	<p>Asserts an existing object as a data object. Use for a data object created by the rule.</p> <p>Multiple asserts result in only one data object instance added to working memory. Multiple creates add multiple data object instances to working memory.</p>	assert employee
assert tree	<p>Asserts a tree of data objects. Specify a data object that is a business object with multiple child objects.</p> <p>Multiple asserts result in only one data object tree added to working memory.</p> <p>In the example, if department is a variable of type Department, and Department has an attribute that's a list of Employee objects, then the department and all its employees are added to working memory.</p>	assert department
assign new	<p>Defines a new local variable and assigns a value to it.</p> <p>To create a variable in a condition, see Using Advanced Conditions.</p>	assign new float Rebate = 200
expression	Performs the functions in the expression.	Duration.years between(Person.birthdate, CurrentDate.date)
return	Stops running rules and returns the output data objects currently in working memory.	return
throw	Throws a preexisting business exception.	throw

Editing Rule and Decision Table Properties

You can change the name, add a description, set the priority, disable, or set an effective date range for a rule or decision table. For a decision table, you can also

determine the policy for handling conflicting rules and decide whether to permit condition combination gaps.

Property	Description
Name	Renames the rule or decision table.
Description	Provides an optional text description of the rule or decision table.
Priority	Values are Lowest, Lower, Low, Medium, High, Higher, and Highest. The default is Medium.
Active	If checked, the rule or decision table is enabled. Deselect to disable the rule or decision table. The default is checked.
Conflict Policy	Specifies how to handle conflicting rules throughout a decision table: <ul style="list-style-type: none"> Manual — You decide which rule overrides another or runs first, and which conflicts to ignore. This is the default. Auto Override — A specific case automatically overrides a more general case. Ignore — Conflicts are ignored. Select this option if all rules can run in any order without causing a problem. See Adding Conflict Resolutions to Decision Tables .
Allow Gaps	If checked, a decision table can omit some possible condition combinations. The default is checked. Check this property if the default path from the exclusive gateway that follows the decision activity in the process can handle the missing combinations. Want to learn more about using an exclusive gateway with a decision in a process? See Assigning a Decision to a Process Component . Deselect this property if the decision table must include all possible condition combinations. To check for missing combinations, select Identify Possible Missing Rules from the Tools menu.
Effective Date	Specifies when a rule or decision table is active: <ul style="list-style-type: none"> Always — Always active, the default. Range — Active between two dates. From — Active after a specified date. To — Active until a specified date. If the Active property is deselected, this setting does nothing.

Adding Conflict Resolutions to Decision Tables

When two or more rules in a decision table have overlapping condition combinations and perform different actions, you must specify how to resolve the conflicts.

Rules *overlap* when at least one of their condition cells has a value in common. Overlap is common when a decision table contains condition cells in which the value doesn't matter (shown as dashes). Overlap without conflict is common and harmless.

Rules are flagged as having a *conflict* when they overlap and they perform different actions. If this happens, you must resolve the conflict.

The Conflict Policy property specifies how to handle conflicting rules throughout a decision table:

- Manual — You decide which rule overrides another or runs first, and which conflicts to ignore. This is the default.
- Auto Override — A specific case automatically overrides a more general case.

For example, suppose a decision table has a condition giving all customers a 10% discount, and another condition giving Gold customers a 20% discount. There is a conflict: both rules apply to Gold customers. However, because Gold customers are a subset of all customers, the Gold customer condition is more specific. To give Gold customers only the 20% discount, you can select Auto Override if this is the only conflict.

- Ignore — Conflicts are ignored. Select this option if all rules can run in any order without causing a problem.

For example, suppose a decision table has a condition giving everyone in the sales department a 3% raise, and another condition giving top performers throughout the company a 5% raise. There is a conflict: which raise do top performers in the sales department get? Auto Override can't resolve the conflict, because neither condition is more specific. To give top salespeople both raises, you can select Ignore if this is the only conflict.

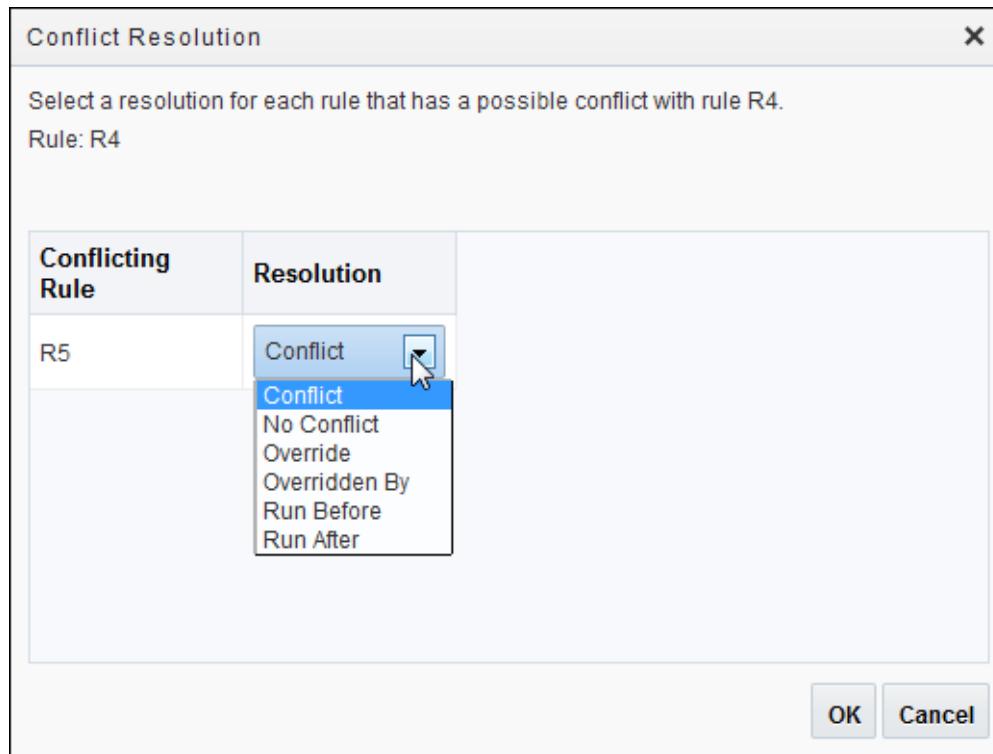
If Conflict Policy is set to Manual, and a conflict occurs, an Unresolved Conflicts row appears in the decision table, in which conflicting rules are listed in each other's columns. You must resolve each conflict separately. You can resolve different conflicts in the same decision table differently.

For example, start with the decision table shown in [Creating a Decision Table](#). Change the rule with income set to [50,000..100,000) and creditGood set to true so that creditGood doesn't matter. In the Unresolved Conflicts row that appears, an R5 link is in the rule R4 column and an R4 link is in the rule R5 column:

	R1 ▾	R2 ▾	R3 ▾	R4 ▾	R5 ▾	R6 ▾
Conditions						
Decision1.in.income ▾	<20,000	[20,000..50,000)	[50,000..100,000)	>=100,000		
Decision1.in.creditGood ▾	-	true	false	false	-	-
Conflict Resolution						
Unresolved Conflicts				R5	R4	
Actions						
create Decision1.out ▾	✓	✓	✓	✓	✓	✓
outcome	"Reject"	"Manual"	"Reject"	"Manual"	"Accept"	"Accept"

This is a real conflict, because a combination of \$50,000–\$100,000 in income and bad credit causes both the Manual and Accept outcomes.

You can click either the **R5** or **R4** link to resolve the conflict. The Conflict Resolution dialog box opens:



Select an option from the Resolution drop-down list:

- Conflict — The conflict remains unresolved.
- No Conflict — The conflict is ignored. Both rules can run in either order without causing a problem.
- Override — The rule in the column header overrides the rule in the cell.
- Overridden By — The rule in the cell overrides the rule in the column header.
- Run Before — The rule in the column header runs before the rule in the cell.
- Run After — The rule in the column header runs after the rule in the cell.

In the example, rule R4 should override rule R5, so that a combination of \$50,000–\$100,000 in income and bad credit causes only the Manual outcome. Therefore, if you clicked the R5 link in the rule R4 column, select Override. If you clicked the R4 link in the rule R5 column, select Overridden By. Either way, the Unresolved Conflicts row changes to Override, and the R4 link in the rule R5 column disappears:

	R1	R2	R3	R4	R5	R6
Conditions						
Decision1.in.income	<20,000	[20,000..50,000)	[50,000..100,000)	>=100,000		
Decision1.in.creditGood	-	true	false	false	-	-
Conflict Resolution						
Override					R5	
Actions						
create Decision1.out	✓	✓	✓	✓	✓	✓
outcome	"Reject"	"Manual"	"Reject"	"Manual"	"Accept"	"Accept"

To achieve the same result if this is the only conflict, set Conflict Policy to Auto Override. Because rule R4 is more specific than rule R5, R4 overrides R5.

For conflicts involving more than two rules, additional considerations apply:

- Override (or Overridden By) lets only one rule run. Override is a combination of prioritization and mutual exclusion. Prioritization is transitive and not symmetric. Mutual exclusion is both transitive and symmetric. If A overrides C and B overrides C, then A or B runs before C but only one of A, B, or C runs.
- Run Before (or Run After) lets all rules run in a prescribed order. Prioritization is transitive but not symmetric. If A runs before B and B runs before C, then A runs before C but B doesn't run before A.

Editing Decision Properties in Business Rules

You can add a decision description, add decision-level actions, and view the input and output data objects.

Property	Description
Description	Provides an optional text description of the entire decision.
Initial Actions	Adds decision-level actions. These run unconditionally, before actions in rules and decision tables run, whenever the decision is invoked in a process. See Using Advanced Actions . For example, if multiple rules and decision tables affect the same output data object, use create in an initial action, then use modify in the rules and decision tables.
Inputs	Displays the following read-only information about each input data object: <ul style="list-style-type: none"> • Name — Identifies the data object in data association. • Business Type — Identifies the data object in rules and decision tables. • List — If checked, the data object can have multiple values. This is the information you supplied when you created the decision. See Creating a Decision .

Property	Description
Outputs	<p>Displays the following read-only information about each output data object:</p> <ul style="list-style-type: none">• Name — Identifies the data object in data association.• Business Type — Identifies the data object in rules and decision tables.• List — If checked, the data object can have multiple values. <p>This is the information you supplied when you created the decision. See Creating a Decision.</p>

12

Integrating Documents and Conversations

Some tasks require more collaboration than others. To help move these tasks along you might need to share documents and have online conversations with other task participants. You can do this by integrating Oracle Content and Experience Cloud with Process.

Topics

- [Why should I integrate documents and conversations?](#)
- [How do I integrate with Oracle Content and Experience Cloud?](#)
- [Roles and Responsibilities](#)
- [Quick Tour of the Documents Page](#)
- [Editing Properties of the Documents Root Folders](#)
- [Defining Folders and Documents](#)
- [Creating a Document- or Folder-Initiated Process](#)
- [Getting Properties for Documents, Folders, and Conversations](#)
- [Editing Conversation Properties](#)
- [Sample Use Case for Documents and Process](#)

Why should I integrate documents and conversations?

Although you can use email to discuss a task and send associated documents, it's much easier to have all that content available while viewing the task and tracking a process instance. Oracle Content and Experience Cloud integrates documents and conversations with your process applications.



What are the benefits of integrating Oracle Content and Experience Cloud:

- **Documents:** Process provides simple file attachment functionality, but if you need something more robust to handle document-intensive processes, you can integrate Oracle Content and Experience Cloud. This service enables you to organize files into folders, manage access to each folder, and even start a process when you upload a document. For example, if you're processing a home loan, you need to manage documents such as loan applications, employment histories, and house appraisals, making sure that the right users see the documents they need to submit, review, or approve, but they don't get access to restricted information.
- **Conversations:** When you integrate conversations, users can easily discuss things that come up during the process. This provides a record of what happened, enabling you to quickly bring new stakeholders up to speed or refer back to things as necessary. Plus, the conversation tools work like the social media tools users regularly use, but with enterprise-wide security and controls. For example, if you're

working on a contract you might need to discuss some of the terms, while still making sure your discussion is confidential.

- **Document- and Folder-Initiated Processes:** You can automatically start a process when someone uploads a document (or folder of documents) to a chosen document folder.

After you integrate Oracle Content and Experience Cloud, document folders and, if enabled, conversations are automatically created for each new application. You can also enable document and conversation functionality in existing applications. You can create additional folders as necessary to further organize documents, setting access to restrict content to the appropriate users. Then your users can upload and view documents and participate in conversations while viewing the task (through icons on the Task Details page).

For a summary of the roles and responsibilities involved in configuring and using Oracle Content and Experience Cloud with Process, see [Roles and Responsibilities](#).

For general information about Oracle Content and Experience Cloud, see [Oracle Content and Experience Cloud Online Documentation Library](#).

How do I integrate with Oracle Content and Experience Cloud?

Before you or any users can access document and conversation features, an administrator must configure settings in both Oracle Content and Experience Cloud and Oracle Process Cloud Service.

Only a user with administrator privileges can establish a connection between the two services.

For Process applications that were created *before* the connection with Oracle Content and Experience Cloud was configured, the document and conversations features are disabled by default. Developers can manually enable these features in their existing applications.

Topics

- [Access Requirements for a Successful Integration](#)
- [Using Documents or Attachments in a Process Application](#)
- [Configuring Settings in Oracle Content and Experience Cloud](#)
- [Configuring Documents Settings in Oracle Process Cloud Service](#)
- [Enabling or Disabling Documents and Conversations](#)
- [Access Issues and Configuration Changes](#)

Access Requirements for a Successful Integration

Note these access requirements for a successful integration:

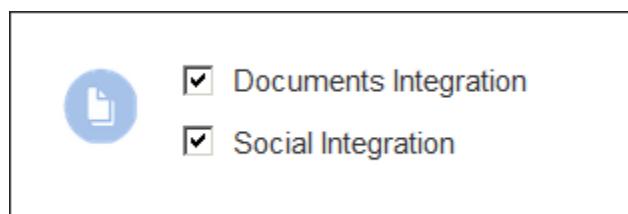
- The Oracle Content and Experience Cloud user configured in Oracle Process Cloud Service must have full access to the folder (that is, the folder of the incoming document) configured in Oracle Content and Experience Cloud to be able to kick off a process.

- For a folder-initiated process (that is, a process with a **Folder Start** event), the Oracle Content and Experience Cloud user configured in Oracle Process Cloud Service must have manager access to that folder in order to access the folder in Oracle Process Cloud Service. For example, manager access is required to see the folder on the Task Details page and the Process Tracking page, access the folder when embedding Process UI components in an external application, or see the folder in the Process Mobile application.
- The Oracle Process Cloud Service user configured in Oracle Content and Experience Cloud must be granted the process initiator role in order to see the process in the process list for a folder and to be able to initiate the process instance upon arrival of a new document.
- To be able to view conversations in Oracle Process Cloud Service, the user must be assigned one of the following roles: Oracle Content and Experience Standard user, Oracle Content and Experience Enterprise user, or Oracle Documents Cloud Service user.

Using Documents or Attachments in a Process Application

Oracle Process Cloud Service automatically includes standard **file attachment** functionality for your process applications. You can upload files and attach them to a process. Now, when you use Oracle Content and Experience Cloud with Oracle Process Cloud Service, you also get **documents** functionality, which lets you upload files, organize files into folders, manage access to each folder, and even start a process by uploading a document. In addition, each process application has the option of using EITHER documents OR attachments.

You control whether a process uses documents or attachments at the application level. Documents are enabled by default. The setting is in the Information pane for the application:



For an application where **Documents Integration** is enabled, then that application can use documents and only documents. For an application where **Documents Integration** is disabled, then that application can use attachments and only attachments. Basically, by disabling documents for an application, you enable attachments for that application.

Remember that the setting applies to the application. Some applications may use documents; others may use attachments. Documents and attachments are mutually exclusive for a given application. You can use one or the other in an application, but not both.

If you use REST APIs to interact with Oracle Process Cloud Service, be sure to use the appropriate API depending on whether your process application uses documents or attachments:

- For documents, be sure to use the /folders REST APIs.

- For attachments, be sure to use the /attachments REST APIs.
If you use the wrong API, then the application will either return an error message or end in a no-operation.

Configuring Settings in Oracle Content and Experience Cloud

In Oracle Content and Experience Cloud, an administrator must enable Oracle Process Cloud Service, enter connection information, and enable Oracle Process Cloud Service use for one or more folders.

Want to learn more about each step? See Integrating with Process in *Administering Oracle Content and Experience Cloud*.

Configuring Documents Settings in Oracle Process Cloud Service

As the administrator, you must also configure the connection between Oracle Process Cloud Service and Oracle Content and Experience Cloud. You need to enter information such as the URL and sign-in credentials for your Oracle Content and Experience Cloud.

To configure the settings in Oracle Process Cloud Service:

1. Go to the Home page, click **Configure**.
2. Click **Services**.
3. In the **Oracle Documents Cloud Service** (Oracle Content and Experience Cloud) section, enter the following information:
 - **URL**: the web address of your Oracle Content and Experience Cloud. Your service administrator receives a Welcome to Oracle Cloud email when the service is ready to use. The email has the URL for your Oracle Content and Experience Cloud. For example, https://your_service_name.com/documents.
 - **Identity Domain**: the name of the identity domain that your Oracle Content and Experience Cloud belongs to. The identity domain is mandatory only if you want conversations as well as documents. You can successfully configure a connection to your Oracle Content and Experience Cloud (for the documents feature) without providing an identity domain.
 - **User and Password**: the account credentials for a user who has access to your Oracle Content and Experience Cloud. This user account is used to test the connection between the services. It's also used during runtime to connect to the services and perform all the runtime operations, such as creating folders and processing conversations.

! Important:

If you want to collaborate using conversations, then the user you specify here must be assigned the following role:

```
service_instance_name OSN_SERVICE_INTEGRATOR_USER_ROLE
```

Use the Oracle Cloud My Services application to assign roles to your users.

4. Click **Test.**

Whenever you make any changes to the configuration settings, it's a good idea to verify that the values you entered are correct. You want to confirm that a successful connection has been established with Oracle Content and Experience Cloud.

Review the test results, which may include messages, errors, and warnings.

5. Select one of the following options to continue:

- If there are any errors or warnings, make the necessary changes and then click **Test** again to verify the new values. Repeat the test each time you change the settings.
- If the connection test is successful, click **Save** to save the configuration settings.
- If you want to cancel and return to the last-saved values, click **Revert**.

Enabling or Disabling Documents and Conversations

After an administrator establishes a connection between Oracle Content and Experience Cloud and Oracle Process Cloud Service, you can use documents and conversations in your process applications. These features are enabled by default for all new applications you create.

For your Oracle Process Cloud Service applications that were created *before* the connection with Oracle Content and Experience Cloud was configured, the documents and conversations features are disabled by default. If you want these features in existing applications, then you need to manually enable the features.

There are many benefits to having documents and conversations available when you're modeling a process, viewing tasks, and tracking a process instance. However, for some particular applications, you might not want the documents or conversations feature to be available. In such cases, you can manually disable one or both features.

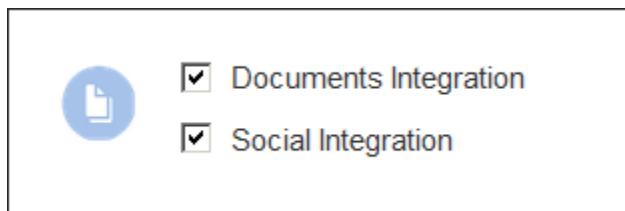
To enable or disable the documents and conversations features for a particular application:

1. Open the application so you can see it on the Application Home tab.

If you're viewing a QuickStart App or a QuickStart Master, then you'll need to click **Switch to Advanced View**.

2. Click **More details  to expand the Information pane.**

The check boxes next to the Documents icon show you the current status of the documents and conversations features for this application.



3. Use the check boxes to enable or disable each feature for this application.

 **Note:**

For any given process application, documents and attachments are mutually exclusive. For applications where **Documents Integration** is enabled, then those applications can use documents and only documents. When you disable **Documents Integration** for an application, then that application can use attachments and only attachments. By disabling documents, you enable attachments.

Access Issues and Configuration Changes

After you configure Oracle Content and Experience Cloud in Oracle Process Cloud Service, keep in mind that access issues or configuration changes can result in errors.

For example:

- If you modify the Oracle Content and Experience Cloud configuration that is currently in use, its associated documents and conversations can no longer be accessed.
 - For documents, you'll get an error when you click **Documents**  or when you access a document from the list on either the Task Details page or the Tracking page.
 - For conversations, you'll get an error when you click **Conversations**  on the Task Details page.
- If an application folder created by a process gets removed, an administrator must restore it from the Oracle Content and Experience Cloud trash folder.

Roles and Responsibilities

The following table provides a summary of the roles and responsibilities involved in configuring and using Oracle Content and Experience Cloud with Oracle Process Cloud Service.

Role	Tasks
Administrator (Runtime)	<p>Configure a connection to Oracle Content and Experience Cloud.</p> <p>You must configure a connection before anyone can use Oracle Content and Experience Cloud features.</p>

Role	Tasks
Developer (Design time)	<ul style="list-style-type: none"> • Manually enable the document and conversation features. To include document and conversation features in applications that were created before you configured the connection to Oracle Content and Experience Cloud, you need to manually enable them. • Create additional document folders. To further organize documents or restrict access to folders that will store important documents, you can create additional folders for a process instance. • Edit the basic properties for root document folders. If you created additional document folders, select which folder is displayed when the process is started. You can also change the default application folder name or the default process instance folder name (for example, if you have a folder naming standard). • Override the default document folder access for a particular task. To change the folder access for a user completing a particular task, you can override the default folder access. For example, you might hide an underwriting documents folder by default (by setting the access to None), but change the access to Contributor during the underwriter's approval task. • Edit the basic properties for the application-level conversation. Control whether or not to automatically post default messages and task assignment messages to the conversation. You can also change the default application conversation name (for example, if you have a conversation naming standard) or add a description to clarify the purpose of the conversation. • Override the default conversation property settings for a particular task. To change the messages that are automatically posted to the conversation for a particular task, you can override the default conversation properties, even adding custom entry and exit messages. • Create document- or folder-initiated processes. To start a process when someone uploads a document (or folder of documents) to a particular document folder, you can use the document start or folder start event to model a process in Composer.
End User (Runtime)	<ul style="list-style-type: none"> • Add a document to a task. Adding a document might complete a task within the process (for example, submitting a loan application or travel request), or it might just provide information to other users that interact with the task (for example, uploading justification for a travel expense). • Manage documents in folders. You can perform actions on files, such as moving or uploading a new version. You can also browse and create folders, including multiple levels. • Collaborate with other task participants. You can share ideas, make decisions, and lend guidance about tasks through online conversations.

Quick Tour of the Documents Page

The Documents page provides the tools required to quickly update the settings and permissions for folders and documents that are stored in Oracle Content and Experience Cloud and used in Oracle Process Cloud Service at runtime.

 **Note:**

The Documents page is available only if an administrator for Oracle Process Cloud Service created a connection to Oracle Content and Experience Cloud.

In addition, for applications that were created *before* a connection to Oracle Content and Experience Cloud was configured, the Documents feature and the Conversations feature are disabled by default. For these cases, you must manually enable the Documents and Conversations features.

Configured Folders and Documents

The Documents page displays information about the folders and documents that have been configured for your application.

For each application that has the Documents feature enabled, Oracle Process Cloud Service automatically creates the following default folders:

- **Application Folder:** This main root folder contains all the process instance folders.
- **Instance Folder:** One root folder is created for each process instance. It contains the set of managed folders that have been defined in Composer.

 **Note:**

The incoming documents and incoming folders that have been defined in Composer aren't stored in the Instance Folder.

- **Managed Folders:** These folders are defined to organize the uploaded files and to set the access level. One managed folder is automatically defined and marked as the Startup folder. The **Startup folder** is the only folder that's shown for users to upload documents when starting an application. You can define additional managed folders. All managed folders are created for each process instance.

Select **Properties** to change the name of the root folders or to specify a different folder as the startup folder.

The screenshot shows the Oracle Application Home interface for a 'Home Loan' document. On the left, a sidebar lists categories: Processes (1), Web Forms (1), Business Types (6), Decisions (0), Connectors (0), and Documents (4). The 'Documents' category is highlighted. The main area displays the document structure under 'Home Loan {instance.id}':

- Applicant Documents:** Description: 'Documents submitted by the person applying for the home loan.' Type: Managed Folder (orange icon). Access: Contributor, checked for Startup.
- Appraisal Documents:** Description: 'Documents submitted by the property appraiser.' Type: Managed Folder (orange icon). Access: Contributor.
- Home Loan Application:** Description: 'Incoming Document.' Type: Incoming Document (orange icon with arrow). Access: Contributor.
- Loan Package:** Description: 'Incoming Folder.' Type: Incoming Folder (orange icon with arrow). Access: Contributor.

At the top right are buttons for Search, Views (grid), New (plus), and Properties. At the top center is a breadcrumb trail: Home Loan > Home Loan {instance.id}. At the top right are navigation icons: a right arrow, three dots, and a plus sign.

Information Displayed for Each Folder or Document

Each property card includes an icon that indicates whether the definition is for a managed folder (), an incoming folder (), or an incoming document ().

Each property card also displays the following information:

- The name of the folder or document. You can click the name to edit the basic properties.
- A description of the folder or document. If no description was provided, then the card displays the definition type: **Managed Folder**, **Incoming Folder**, or **Incoming Document**.
- The access type displays the default permission level required to access the folder or document. You can override the default access type at the task level. The available access types are Contributor (default), Downloader, Viewer, or None.

Note that **Startup ✓** indicates which folder has been selected as the Startup folder. Select **Properties** to select a different folder as the startup folder. The Startup folder is the only folder that is shown when a process is started.

Summary of Tools for Managing Folders and Documents

Use the following tools to view, configure, and manage your folders and documents:

- Click **Views**  to alternate between viewing the page in either a grid format or a list format.
- Click **New Folder or Document**  to configure folders and documents.
- Click **Properties** to edit the name of the root folders or to change the startup folder.

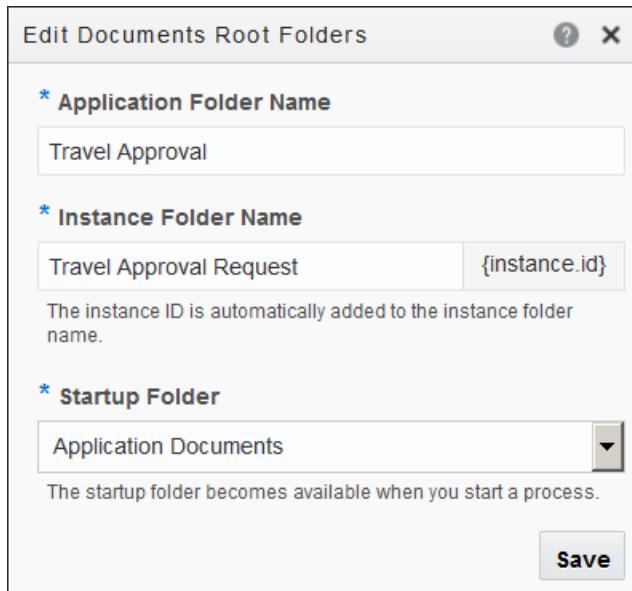
- Click the folder name to edit the basic properties of the folder or document.
- Click **Delete**  to delete a folder or document.

Editing Properties of the Documents Root Folders

For each application that has the Documents feature enabled, Process automatically defines an application folder, a process instance folder, and a managed folder. You can change the name of the root folders or specify a different folder as the startup folder.

To edit the basic properties of these folders:

1. Select your process application.
2. Click **Documents**.
3. Click **Properties**. The Edit Documents Root Folders dialog box opens.



4. Make any necessary changes.
 - **Application Folder Name:** This folder is the main root folder for the application. It contains all the process instance folders.
 - **Instance Folder Name:** One root folder is created for each process instance. It contains the set of managed folders that have been defined in Composer.

 **Note:**

The incoming documents and incoming folders that have been defined in design time aren't stored in the Instance Folder.

- **Startup Folder:** The folder that you select is the only folder that's shown for users to upload documents when starting an application.
5. Click **Save**.

Defining Folders and Documents

When you're designing a process application, you can create definitions for managed folders, incoming folders, and incoming documents for each process instance. You use managed folders to organize files. You create definitions for incoming folders or documents if you plan to model a process that uses a folder start event or a document start event.

For managed folders, incoming folders, and incoming documents, you can define the access level (also called permission), and override the default permission at the task level. For example, you can restrict access to important documents that are required during the business process.

To configure folders and documents:

1. Select your process application.
2. Click **Documents**.
3. Click **New Folder or Document** .
4. Select whether you want to create a definition for a managed folder, an incoming folder, or an incoming document.
5. In the **Name** field, enter a unique name.
 - If you're defining a managed folder, the folder gets created with this name when the process instance gets initiated.
 - If you're defining an incoming folder, you use this name to associate the folder with a folder start event.
 - If you're defining an incoming document, you use this name to associate the document with a document start event.
6. In the **Description** field, enter a brief explanation that can help other users understand the purpose of the folder or document.
7. In the **Default Access Type** field, select one of the following roles from the drop-down list:
 - **Contributor**: Contributors have complete access. They can modify, update, upload, delete, download, save, and view files.
 - **Downloader**: Downloaders can download files and save them to their own computer. They can also view folders and files, but can't change things.
 - **Viewer**: Viewers can only look at folders and files. They can't change things.
 - **None**: No access allowed.

The role you select defines the default access. You can then [override the default access](#) at the task-level, if required.

8. Click **Create**.

The new folder or document appears on the Documents page.

To edit the basic properties of a folder or document, go to the Documents page and click the name of the folder or document.

Creating a Document- or Folder-Initiated Process

Use the **Document Start** event to model a process that can be initiated by a document. Use the **Folder Start** event to model a process that can be initiated by a folder.

You must have an Oracle Content and Experience Cloud, and you must configure a connection between that service and your Oracle Process Cloud Service before you can create a document-initiated or a folder-initiated process.

By enabling the Oracle Content and Experience Cloud integration, you can define folders that will be created automatically on Oracle Content and Experience Cloud for every process instance, providing a predefined organization of the documents involved. You can also override the access type at the task-level to define the right permissions for that folder or document for a particular task based on your business needs. For example, you might want to prevent users from viewing a classified document or folder associated with a task.

Note these access requirements for a successful integration:

- The Oracle Content and Experience Cloud user configured in Oracle Process Cloud Service must have full access to the folder (that is, the folder of the incoming document) configured in Oracle Content and Experience Cloud to be able to kick off a process.
- For a folder-initiated process (that is, a process with a **Folder Start** event), the Oracle Content and Experience Cloud user configured in Oracle Process Cloud Service must have manager access to that folder in order to access the folder in Oracle Process Cloud Service. For example, manager access is required to see the folder on the Task Details page and the Process Tracking page, access the folder when embedding Process UI components in an external application, or see the folder in the Process Mobile application.
- The Oracle Process Cloud Service user configured in Oracle Content and Experience Cloud must be granted the process initiator role in order to see the process in the process list for a folder and to be able to initiate the process instance upon arrival of a new document.

To design a process that can be started by a document or folder:

- Define the **Incoming Document** or the **Incoming Folder** in Oracle Process Cloud Service.
- Model a process that has a document start event or a folder start event.
- Customize its implementation to map the start event with the corresponding incoming document or folder you created. Implementation options allow you to define the way in which the document or folder is exposed to users.
- Modify what role can access the document or folder at the task level (optional).
- Configure the folder on Oracle Content and Experience Cloud to initiate a process on document arrival.

Alternatively, you can use REST APIs to instantiate a process instance and provide all the input values.

Defining the Incoming Document or Folder

To define the incoming document or folder, open the process application you are modeling and click **Documents**. The Documents page lists the incoming documents and incoming folders that have already been defined.

To create a new incoming document or folder, click **New** , select the appropriate type, enter a name, and select the default access (permission).

Modeling Processes with Document or Folder Start Events

Next, build your process that has a document start or folder start event. In the current release, there are some limitations for how you can create the process. Failure to follow the required procedure will invalidate the process.

Caution:

You must add the document start or folder start event from the Elements palette. Only a single, and only the first, start event that you add to a process will be supported. Multiple start events aren't supported in this release. Also, don't change a start event using the Change Type option, and don't delete the first start event. Both actions will invalidate the process for use.

To model a process that has a document start or folder start event:

1. On the Application Home tab, click **Processes**.
2. Click **New process**  to open the Create Process dialog box.
3. Select **None**.

Note:

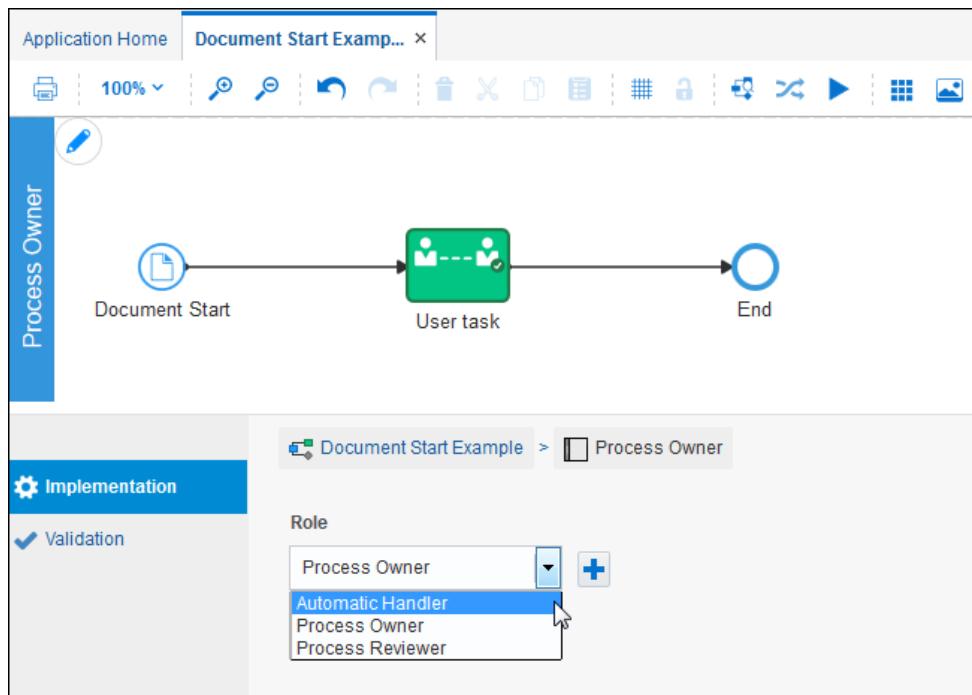
Be sure to select **None**. You will change this empty start event later.

4. Enter a name for the process, confirm the **Open Immediately** check box is selected, and then click **Create**.
The canvas displays the start and end events in the process.
5. Click **Events** in the Elements Palette.
6. Drag either the **Document Start** event or the **Folder Start** event to the canvas.
7. Add a sequence flow from your new document or folder start event to the end event.
8. Delete the empty start event.
9. Continue to modify and define the process. Be sure to test and deploy the process.

Allowing All Users to Start a Document- or Folder-Initiated Process

To allow any valid user to upload a document to start a process:

1. Open the process.
2. In the swimlane with the document or folder start event, click the role name and then click **Edit** .
3. In the **Role** field, select **Automatic Handler**.



Customizing Your Document- or Folder-Initiated Process

After you model the process, you can add a start document or folder in a process and customize it.

In your process, click **Document Start** or **Folder Start** event in the process diagram.

From the icons that display, click **Menu**  and then select **Open Properties**. The Properties pane expands into view below your process diagram.

Define how you want to handle the incoming document or folder:

- **In Place:** Selecting this value keeps the location of the document as it is wherever it is. Optionally, you can map the incoming document or folder with one of the predefined documents or folders (open the drop-down list) for its management.
If there isn't a predefined incoming document or folder suitable for this particular process, then click **New**  to create one.
- **Unmanaged:** The document or folder is ignored by the current process. Oracle Process Cloud Service won't show an unmanaged document or folder in runtime. It's up to the process modeler to handle the incoming document or folder. For example, if you want to move the incoming folder or document to another location, then you can use an XPath expression to get properties, such as `Id` or `Type`, and pass the property information to a REST service.

Customizing Access at the Task Level

In your process, click a user task in the flow diagram. From the icons that display, click  and then select **Open Properties**.

The Properties pane expands into view below your process diagram.

Click **Override** to customize the current permission of any element defined.

Configuring the Folder

To have a process automatically start when a document is uploaded to a folder, you need to configure the folder on Oracle Content and Experience Cloud to initiate a process on document arrival:

1. Sign in to Oracle Content and Experience Cloud.
2. Select the folder.
3. Select **Properties** from the menu bar.
4. Enable the **Initiate process on document arrival** setting.
5. Select the process from the list.
6. Click **Save**.

The folder and its subfolders are now available for use within the Oracle Process Cloud Service interface. Any change to a file in the folder or any new file uploaded to the folder triggers the process associated with the folder. You can override the inherited process for a subfolder, but you can't disable the association with a process.

When a file is uploaded from Oracle Content and Experience Cloud into a folder configured for use with Oracle Process Cloud Service, the file is used for the task associated with that folder. Users in Oracle Process Cloud Service can take any actions on the files there, such as approvals. When a task step is completed, the file can be moved or managed according to the defined process.

When Oracle Content and Experience Cloud starts a process, the payload sent to launch the process includes this information:

- Document ID
- Document name
- ID of the user who started the process
- Type
- Role (indicates the role that should be used to generate subsequent application links)
- Version

The following example uses only the document ID and document name for display in the form in Oracle Process Cloud Service. In addition, the document ID is used for making REST API calls to move or copy the file in Oracle Content and Experience Cloud into the task folder.

```
{  
  "processDefId": "testing~UserFileApproval!1.0~FormApprovalProcess",  
  "operation": "startEvent",  
  "params": {
```

```
        "id": "D2806600E495B744E66BF3981212FF6185DE89BE6812",
        "type": "d",
        "name": "document-name",
        "startedBy": "user-id",
        "role": "role that should be used to generate subsequent applinks",
        "version": "version"
    }
}
```

 **Note:**

The first operation in the Oracle Process Cloud Service process must be a start event. Otherwise, Oracle Process Cloud Service will return a 500 error to Oracle Content and Experience Cloud.

As a developer, you must be aware of the following requirements for the process you develop:

- It needs to be a process that uses an Oracle Content and Experience Cloud Start event.
- When deploying the process, you need to share it with the user specified for enabling the integration so that user has the rights to trigger the process.
- For the user who uploaded the file to show up as the user who started the task, the process must use the value passed in the **startedby** field as the display name for the initiator.
- If you enable the process integration for a folder, you need to share this folder with the Oracle Content and Experience Cloud user that was used to enable the integration in Oracle Process Cloud Service.

Using REST API Calls to Instantiate a Process

Most times, you will want to use the document start event or the folder start event to model a process that can be initiated by a document or folder. Alternatively, you can use REST APIs to instantiate a process instance and provide all the input values.

Fetching Processes Started by Documents or Folders

To fetch only those processes that can be started by a document, use the **doc** value in the **interfaceFilter** filter for the process-definitions API.

```
GET /bpm/api/3.0/process-definitions?interfaceFilter=doc
```

To fetch only those processes that can be started by a folder, use the **folder** value in the **interfaceFilter** filter for the process-definitions API.

```
GET /bpm/api/3.0/process-definitions?interfaceFilter=folder
```

As with any process, Oracle Process Cloud Service doesn't validate if a process instance has already been initiated for the same set of input arguments. An end user can instantiate any number of process instances for the given document or folder details.

Any user who has the permission/role to instantiate this process can invoke the above API to query and fetch the list of processes that can be started by a document or folder.

Instantiating a Process Instance

To instantiate a process instance, including a document-initiated or a folder-initiated process, use the standard API:

```
POST /bpm/api/3.0/processes
```

You pass several parameters that define the ID, type, and name of the document or folder.

For example:

```
{
  "processDefId": "process-definition-id",
  "operation": "startEvent",
  "params": {
    "id": "document-id|folder-id",
    "type": "d|f",
    "name": "document-name|folder-name",
    "startedBy": "user",
    "role": "contributor"
  }
}
```

For the `type` input parameter, you can enter "`d`" for document or "`f`" for folder.

Currently, the `role` input parameter is ignored. All process participants automatically get contributor role access to the document or folder that initiated the process instance.

Like all processes, only users who are members of the start swimlane role can instantiate a process.

Sample Code

You can use the `cURL` command to invoke the Process REST API. For example:

```
curl.sh
host=localhost
port=7001
user="user_name:password"
curl -H "Content-Type:application/json" -u $user -d @input.json
http://$host:$port/bpm/api/3.0/processes
```

Here is sample `input.json` that adheres to the required interface. Note that for the `type` parameter, a "`d`" indicates document and an "`f`" indicates folder.

```
input.json
{
  "processDefId": "default~DocumentStart!2.0~Process1",
  "operation": "startEvent",
  "params": {
    "id": "document-id|folder-id",
    "type": "d|f",
    "name": "document-name|folder-name",
    "startedBy": "user",
    "role": "contributor"
  }
}
```

```
        }  
    }
```

And here is the sample response returned by this API:

```
JSON response  
{  
    "levels": 0,  
    "links": [  
        {  
            "href": "http://adc01jkn.us.oracle.com:7001/bpm/api/3.0/processes/19",  
            "length": 0,  
            "rel": "self"  
        },  
        {  
            "href": "http://adc01jkn.us.oracle.com:7001/bpm/api/3.0/processes/19/audit",  
            "length": 0,  
            "rel": "audit"  
        }  
    ],  
    "title": "Instance #19 of Process1",  
    "processId": "19",  
    "processDefId": "default/DocumentStart!2.0/Process1g",  
    "processName": "Process1",  
    "priority": 0,  
    "owner": "DocumentStart.ProcessOwner",  
    "creator": "mynname",  
    "state": "OPEN"  
}
```

Getting Properties for Documents, Folders, and Conversations

You can use an XPath expression to fetch document properties, folder properties, or conversation properties given the name of the document, folder, or conversation. You can then pass the property information to a REST service.

For example, as part of your document workflow patterns, you may have a need to copy documents *to a process folder* or get documents *from a process folder*. Suppose you have a process that manages transactional content and documents from a central IN box initiate the process. Once the process is initiated, the documents need to be moved from the central IN box to a process folder.

Or, after documents have been successfully reviewed and approved, you might want to move those documents to a final approved location. For example, think about a process that manages market assets. In this case, you might want to get the documents from a process folder and move them to an approved marketing collateral folder.

Get Document Asset Property

In the Expression Editor, you can select **Get Document Asset Property** from the Document Service option. You can get a property for an asset, which in this case can be a document or a folder.

Here is the usage format:

```
DocumentService.getDocumentAssetProperty(<propertyName>, <documentAssetName>)
```

You need to enter two parameters:

- Property Name, which is of String type, defines the property that you want to get. Supported properties are `Id` and `Type`. The value returned for `Id` is the actual ID of the document or folder used in Oracle Content and Experience Cloud. The value returned for `Type` is `INCOMING_DOCUMENT`, `MANAGED_FOLDER`, or `INCOMING_FOLDER`.
- Document Asset Name, which is the name of the document or folder.

The command returns a property value of string type.

If you don't provide a parameter or if you provide more than two parameters, an error occurs.

Get Conversation Property

In the Expression Editor, you can select **Get Conversation Property** from the Social Service option.

Here is the usage format:

```
SocialService.getConversationProperty(<propertyName>, <conversationName>)
```

You need to enter two parameters:

- Property Name, which is of String type, defines the property that you want to get. The supported property is `Id`. The value returned for `Id` is the actual ID of the conversation used in Oracle Content and Experience Cloud.
- Conversation Name, which is the name the user defined for the conversation.

The command returns a property value of string type.

If you don't provide a parameter or if you provide more than two parameters, an error occurs.

Defining the Simple Expression in the Editor

To get properties and access the ID for folders, documents, or conversations:

1. Open Composer, and then open an application.
2. Click **Processes** in the Components pane.
3. Open a process, and click an activity in the process.
4. Click Menu , and select **Open Data Association**.
5. Click **Expression Editor** .
6. Select the **Operators** tab.

In the Expression Editor dialog box, locate the following options on the Operators tab:

- Document Service
 - Social Service
7. Select the expression to insert.

As with all simple expression functions, you click **Insert into Expression** to enter the parameter values and then click **Validate** to verify your expression.

An empty string is returned if there are no values for the document, folder, or conversation properties.

Example XPath Expression	Sample Result
DocumentService.getDocumentAssetProperty("Id", "TestFolder")	F14573D5EECE5BAB5724CE41T0000DEFAULT00000000
DocumentService.getDocumentAssetProperty("Type", "TestFolder")	MANAGED_FOLDER
DocumentService.getDocumentAssetProperty("Id", "Incoming_Document")	D214D537E68C726FB70FF6E1T0000DEFAULT00000000
DocumentService.getDocumentAssetProperty("Type", "Incoming_Document")	INCOMING_DOCUMENT
SocialService.getConversationProperty("Id", "DCSApplication")	307703307

Editing Conversation Properties

Conversations are beneficial for process flows that contain manual tasks where task participants are required to collaborate at different levels to get guidance, make decisions, and so on.

Conversations provide task participants the ability to consult with people of authority to get guidance and help make decisions, as well as the ability to collaborate with other task participants to review and clarify data and document related information.

! Important:

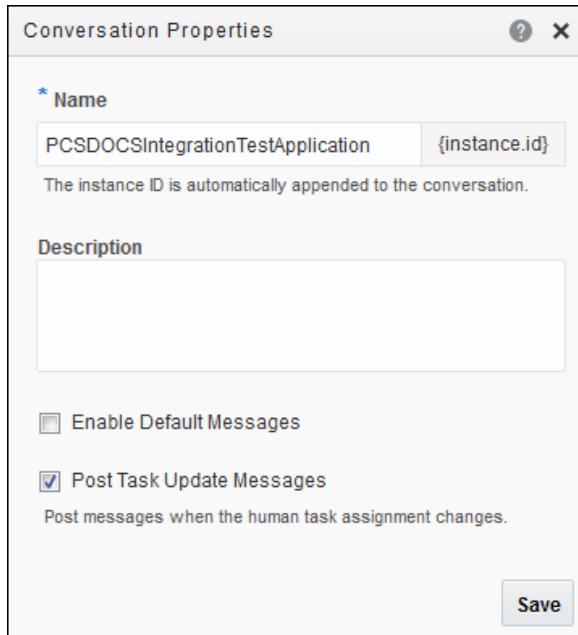
Before you can use conversations in Process, the administrator must create a connection to Oracle Content and Experience Cloud. Want to learn more about how to create this connection? See [How do I integrate with Oracle Content and Experience Cloud?](#)

Once a connection has been created, the system automatically creates a conversation for each new application instance. For example, the system creates a conversation for each new travel request.

Process developers can set general conversation properties at the application level and also at the activity level for each activity. For example, for a task activity, you can have messages automatically posted to the application-level conversation on task instance creation and completion.

To set general conversation properties for the application:

1. Go to the Application Home tab, and then click **Conversation**. The Conversation Properties dialog box opens.



2. Enter the following information:

Field or Check Box	Description
Name	Provide a unique name for this application conversation. Note that the instance ID is automatically appended to the conversation name you specify.
Description	Provide a short description of one or two sentences to help distinguish this conversation from others with a similar name.
Enable Default Messages	Select this check box if you want default messages posted when a process starts and when it completes, and if you want default entry and exit messages posted for each activity executed.
Post Task Update Messages	Select this check box to post messages to the conversation any time a task is updated. These updates include any changes to the human task assignment or any change in the task status such as suspended, resumed, escalated, acquired, released, renewed, delegated, updated, added comment, requested info, and submitted info.

3. Click **Save**.

 **Note:**

For applications that were created *before* a connection to Oracle Content and Experience Cloud was configured, the Conversations feature is disabled by default. For these cases, you must manually enable the Conversations feature. To do so, go to the Application Home tab, open the Conversation Properties dialog box, and click **Enable**.

Sample Use Case for Documents and Process

Documents are an integral part of Oracle Process Cloud Service because they're commonly used in approval flows. Let's look at a sample process for getting a home loan.

During the loan process, different types of documents must be filed and reviewed, including:

- Applicant related documents, such as bank statements, pay stubs, W-2 form, and credit reports
- Property related documents, such as title and appraisal

Different roles, such as loan officer, loan processor, and loan underwriter, review the documents and approve the loan. Once the loan is approved, additional documents, such as loan agreement documents, must be created, signed, and approved.

These review and approval activities require segregating documents into different folders, moving them to approved folders, and so on, during the loan processing flow. Integrating Oracle Content and Experience Cloud with Oracle Process Cloud Service provides the capability for you to create and manage these folders in design time.

Document Folders

Using the home loan as an example, the following table lists the folders that would be required for the loan process and the types of documents you would expect to find in each folder.

Folder	Description
Applicant Documents	Documents related to the applicant, such as W-2 form and bank statements
Appraisal Documents	Documents related to the appraisal, such as property photos and recent comparable sales
Approval Documents	Documents related to the loan approval, such as credit score, risk model analysis, and loan term computation
Agreement Documents	Documents containing the loan terms and conditions
Property Documents	Documents related to the property, such as title and insurance
Funding Documents	Documents created as part of funding, such as updated title and wiring instructions

Actions Performed by Tasks on Folders

Continuing with our home loan example, the following table shows the sequence of tasks in the loan process, lists who performs the task (swimlane), and describes the task performed on the folder.

Sequence	Swimlane	Task
1	Loan Applicant	Uploads W-2 forms and bank statements to the Applicant Documents folder.
2	Property Appraiser	Uploads appraisal related documents to the Appraisal Documents folder.

Sequence	Swimlane	Task
3	Loan Agent	Reviews both the applicant documents and the appraisal documents to ensure completeness. If additional documents are required, then request the applicant or appraiser to upload them.
4	Loan Officer	Reviews both the applicant documents and the appraisal documents, approves the loan, and creates loan documents in the Agreement Documents folder. The applicant documents, appraisal documents, and approval justification documents are stored in the Approval Documents folder.
5	Loan Applicant	Signs agreement documents and uploads them to the Agreement Documents folder.
6	Title Company	Uploads all documents related to the property to the Property Documents folder.
7	Finance Officer	Reviews approval documents, agreement documents, and property documents in their respective folders, and funds the loan. Also uploads related documents to the Funding Documents folder.

Folder Permission for Various Tasks

Finally, for the home loan example, the following table lists the access permission required for each folder based on the user and the task.

Swimlane	Tasks	Folder and Access Permission
Loan Applicant	Submits applicant documents, and if approved, submits agreement documents	Applicant Documents and Agreement Documents folders; Contributor access
Property Appraiser	Submits appraisal documents	Appraisal Documents folder; Contributor access
Loan Agent	Reviews applicant documents	Applicant Documents and Appraisal Documents folders; Viewer access
Loan Officer	Approves the loan	Applicant Documents and Appraisal Documents folders; Downloader access Approval Documents and Agreement Documents folders; Contributor access
Title Company	Submits property documents	Property Documents folder; Contributor access
Finance Officer	Funds the loan	Approval Documents folder; Viewer access Agreement Documents and Property Documents folders; Downloader access Funding Documents folder; Contributor access

Integrating with Applications and Services

You can integrate Process with other applications and services in a number of ways.

Topics

- [What You Can Do on the Integrations Page](#)
- [Working with Integration Cloud Service Integrations](#)
- [Creating REST and Web Service Connectors](#)
- [Applying Message Security to Integrations](#)
- [Invoking ICS Integrations, REST, and Web Services](#)
- [Embedding Process UI Components in Other Applications](#)

Process contains an optional out-of-the-box integration with Oracle Content and Experience Cloud for document and conversation sharing. See [Integrating Documents and Conversations](#).

What You Can Do on the Integrations Page

Use the Integrations page to manage ICS integrations, web service and REST connectors, and to manage the life cycle of web service definition files.

The Integrations page is divided into two views:

- [Integrations](#)
- [Definitions](#)

Use the view option to alternate between views:

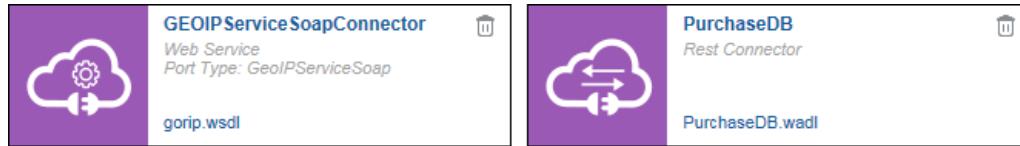


Integrations View

Use the Integrations view to perform the following tasks:

- [Create ICS integrations](#)
- [Create web service connectors](#)
- [Create REST connectors](#)
- View connector information in either Grid or List format. Click **Views**  to alternate between views
- View and edit ICS integrations, and web service and REST connector information
- Delete integrations, and web service and REST connectors

You can see the ICS integrations, and the web service and REST connectors that are available for use in any process created for the selected application, as shown in the Grid view.



To edit a REST or web service connector, click its name. To view the details of the WSDL file, click its name.

Definitions View

Use the Definitions view to perform the following tasks:

- Import web service definitions.
- View WSDL file information either in Grid or List format. Click **Views** to alternate between views.
- View details of the WSDL file, as shown here.

The image shows the 'Definitions View' for the 'gorip.wsdl' WSDL file. It displays a hierarchical list of elements and types defined in the WSDL. At the top is the port type 'GeolPServiceSoap' which maps to a 'ServiceConnector'. Below it are several elements: 'GeolP', 'GetGeolP', 'GetGeolPContext', 'GetGeolPContextResponse', 'GetGeolPResponse', and 'GeolP'. Each element is preceded by a small icon indicating its type (Element or Type).

- Upload a new version of the WSDL file.
- Add a new web service connector for a specific port type.
- Delete WSDL files.

In this view, you see the WSDL files that are included in the application.



Click **Edit** to display a detail view for the web service definition. In this view, a list with the elements of the WSDL is displayed. You can perform the following tasks:

- Edit the web service connector defined for the specific port type.
- Add a web service connector based on the WSDL file and specific port type.

- Upload a new version of the WSDL file.
- Delete a WSDL file.

Note that you can't delete WSDL files that are being used by connectors.

See [Working with Web Service Definition Files](#).

Working with Integration Cloud Service Integrations

ICS Integrations enable users to communicate with local and remote applications that are exposed as either SOAP or REST.

Topics

- [About ICS Integrations](#)
- [Configuring ICS Integrations](#)
- [Editing ICS Integrations](#)

About ICS Integrations

Integration Cloud Service Integrations enable you to communicate with applications in the cloud using adapters.

Use integrations to integrate between diverse systems or endpoints. An integration is composed of two different connections, the source and the target, and a mapping set between the types used by the distinct systems. The target connection is the endpoint system, which is called and is implemented by an adapter. The source connection is the entry point to the integration from Process. It's represented by a source system or endpoint, which triggers the integration execution.

See Getting Started with Oracle Integration Cloud Service in *Using Oracle Integration Cloud Service*.

Process applications can communicate and exchange data with local and remote applications that are exposed as either SOAP or REST.

Once you have set up an ICS integration, you can select it and use it in a process. After setting up an ICS integration, you must map a data object to hold the information to be sent or received as a payload. See [Configuring ICS Integrations](#).

Administrators can customize the service configuration during deployment time, isolate development and production security data, and support scenarios where customers have different servers for testing and production.

You'll need to know the details of the integration in order to use it to implement your service task.

ICS Integrations Setup Overview

To create an ICS integration, you'll need the following information:

- The ICS integration that you want to use to implement your service task
- The user credentials, if the integration was exposed enforcing a security policy
- Optionally, the read timeout and ICS integration connection timeout in milliseconds

Configuring ICS Integrations

Use ICS integrations to communicate with applications using adapters.

 **Note:**

You can create an ICS integration only if you have configured a connection between Oracle Process Cloud Service and Oracle Integration Cloud Service. See [Configuring Oracle Integration Cloud Service](#).

Configuring an ICS integration involves the following main tasks:

- Add a connection to an active ICS integration
- Apply authentication to the ICS integration
- Use the integration in a process

To configure an ICS integration:

1. Add a connection to an active ICS integration.
 - a. Go to the Application Home tab.
 - b. Click **Integrations** and select the **Integrations** view.
 - c. Click **New Integration**  and select **Create ICS Integration**.

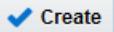
Create ICS Integration

Select Advanced

Select an active Integration Search 

 My Create Org In Service Cloud For Sales Acct <i>My integration to create organization in Service Cloud for Account Creation in Sales</i>	01.00.0000
 My Get Weather Data Service <i>Retrieve weather data by zipcode</i>	01.00.0000
 My Stock Service <i>A demo integration to retrieve stock service</i>	01.00.0000

* Name
MyCreateOrgInServiceCloudForSalesAcct

 Next  Create

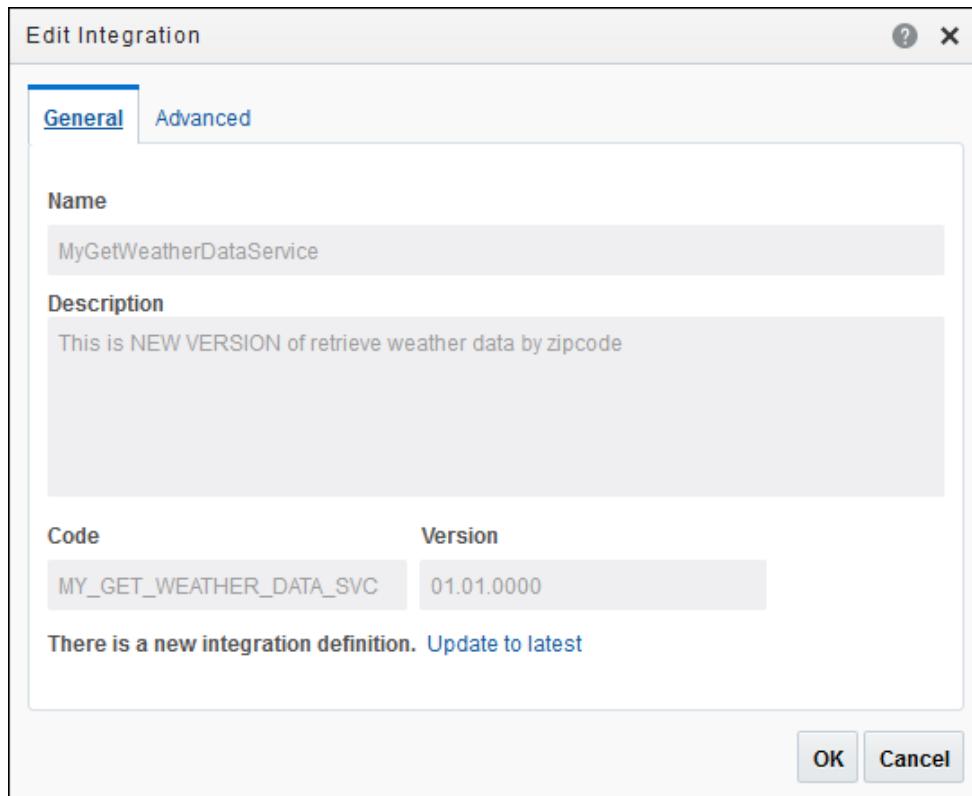
- d. In the Create ICS Integration dialog box, select an ICS integration from the list of active integrations and click **Next**.
2. Apply authentication for the ICS integration.
 - a. Optionally, specify the read timeout and the timeout for connecting to the ICS integration in milliseconds.
 - b. Select the type of security from the drop-down list, and enter the information required for the selected security.
 - Select an existing key from the drop-down list. Note that the user name and password values are automatically populated. If you don't want to use an existing key, then select **[New Key]** and enter a name, user name, and password.
 - If you set the Security to **APP Id — Username Token With Message Protection**, you must also select **New Certificate Alias** from the drop-down list and click **Add Certificate** to upload a certificate.
- Need more information about adding certificates? See [Managing Security Certificates during Design Time](#) and [Applying Message Security to Integrations](#).
- c. Click **Create**.
 3. Implement the ICS integration in one or more service tasks in the process.
 - a. Add or open a service task in the process. In its properties, implement a service call to an ICS integration. See [Invoking ICS Integrations, REST, and Web Services](#).
 - b. Configure data association to pass new values as input and output to the service task. For example, if you're querying an order from Oracle Sales Cloud, pass an Order ID as input and you'll receive an object representing an order as output. You need to map this order object to your business data objects as needed.
 - c. Repeat these steps if you need to configure other service tasks to call the ICS integration, perform an operation, and store data changes in the business object.

To delete an ICS integration, click its delete icon on the Integrations page. You can't delete an ICS integration that is being called in a process.

Editing ICS Integrations

You can edit an existing ICS integration in the Edit Integration dialog box in Integrations view.

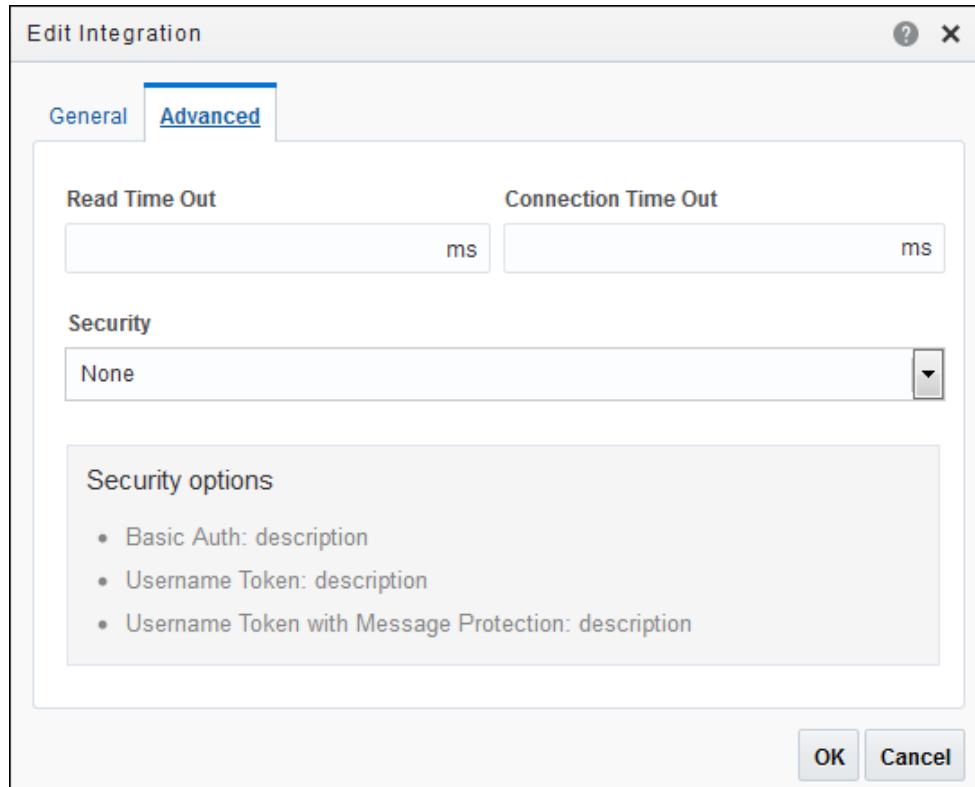
1. Click an integration to open the Edit Integration dialog box.
2. If a new definition of the selected ICS integration version, or a new version of the selected ICS integration is available, a message with a link to update to the latest definition or version appears in the General tab of the Edit Integration dialog. Click **Update to latest** to update to the latest version.



Note:

To view the definition of an ICS integration, on the Application Home tab, click **Integrations**, and then click **Open Definition** for that integration.

3. In the Advanced tab, modify the options as required for the selected ICS integration.



Creating REST and Web Service Connectors

You can create connections to exposed REST and web services using Process. Your process applications can communicate and exchange data with these services.

Topics

- [About REST and Web Services](#)
- [Creating a REST Connector](#)
- [Creating a Web Service Connector](#)
- [Working with Web Service Definition Files](#)

About REST and Web Services

Process applications can communicate and exchange data with local and remote applications that are exposed as either REST or web services.

When considering whether to use REST or web services, keep in mind that some applications support one and some the other, so the protocol decision may already be made for you. As a general rule, use REST services for integration over the web, and use web services for enterprise application integration scenarios. Cloud applications provide open REST APIs for consumers to interact with them and applications running in the cloud typically communicate through REST calls.

REST and web services are client and server applications that communicate over the World Wide Web (WWW) using HyperText Transfer Protocol (HTTP).

For general information about REST and web services, see:

- Oracle Process Cloud Service REST API
- The W3C home page for the Web Application Description Language (WADL) file:
www.w3.org/Submission/wadl
- The W3C home page for the Web Services Description Language (WSDL) file:
www.w3.org/TR/wsdl
- The Web Services chapter of the Java EE 6 Tutorial:
docs.oracle.com/javaee/6/tutorial/doc/gijti.html

You will need some basic information about the REST or web service you want to connect to. However, you don't need to know the details of how services are structured.

Connection to a REST Service

To create a connection to a REST service, you need the following service information:

- Definition of the REST service to connect to WADL, RAML, YAML or other
- URLs to the location of the different resources
- Access to these URLs to get the JSON sample used to create the types needed to send and receive data to/from the service
- List of operations to use on each resource
- List of parameters to pass to operations
- For a secure REST service, the user name and password required to access the service

Need help obtaining this information? See Oracle Process Cloud Service REST API.

Connection to a Web Service

To create a connection to a web service, you need the following service information:

- Location of the WSDL file, as a local file path or a URL
- Port type and callback port type in the WSDL file to select
- For a secure web service, the user name and password required to access the service

Creating a REST Connector

Use the outbound REST connector to call a REST service to retrieve, create, update, or delete data on a web server that supports the REST architecture. The connector enables Oracle Process Cloud Service to interact with other Oracle Cloud applications via REST, including SaaS and PaaS applications running inside or outside Oracle Cloud. Using the REST Connector editor, you define the resources and operations needed to connect to a REST service, regardless of the description language used to define the service.

Creating a REST connector involves the following main tasks:

- Create the connector, which opens the editor.
- Set connector timeout settings.

- Apply authentication to the connector (optional).
- Add resources to the connector.
- Add operations to resources.
- Specify request and response parameters for operations.
- Implement the REST connector within a process.

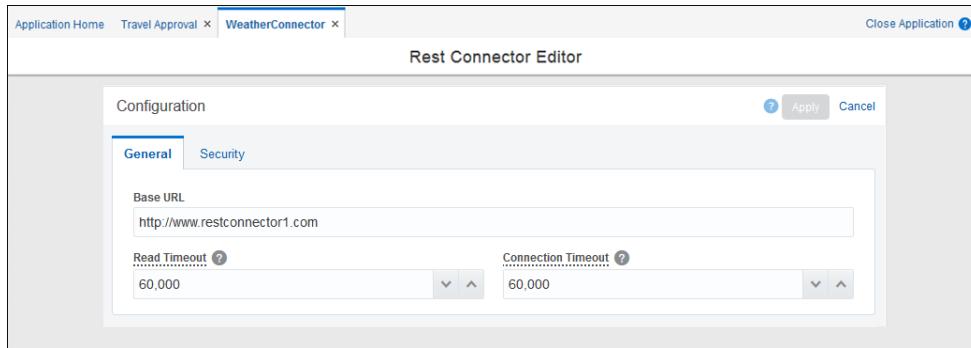
For example, you might create an Oracle Process Cloud Service application that interacts with Oracle Eloqua Marketing Cloud Service via REST service calls, in which the application retrieves an email from Eloqua and updates the email with information from a process.

- The REST connector uses a GET operation to retrieve email content from Eloqua, and a PUT operation to update the email with the response. Business objects contain the sample payload values.
- The process contains two service tasks that act as Eloqua integration points. The process receives the email ID to approve. One service task fetches the email content from Eloqua, using the supplied email ID. After an approval or reject action, the other service task modifies the email content to contain the outcome of APPROVED or REJECTED.

To create, configure, and implement a REST connector:

1. Create a REST connector.
 - a. On the Application Home tab, click **Integrations** and select the **Integrations** view.
 - b. Click **New Integration** , click **External**, and then select **REST**.
 - c. In the Create REST Connector dialog box, enter a name and base URL for the REST service, and click **Create**.
The Rest Connector editor opens, with expandable Configuration and Resources sections. The Configuration section displays the base URL you specified.
 - d. Click **Edit** to expand the connector's configuration settings.
2. Set timeout settings for the REST connector (optional).
 - a. In the **Read Timeout** field, specify the timeout in milliseconds to wait to read data from the host.
 - b. In the **Connection Timeout** field, specify the timeout in milliseconds to make the initial connection.
3. Apply authentication to the REST connector (optional).

By default, no security is configured. You can apply HTTP basic authentication, which provides access control to web resources by requiring a user name and password when making requests.



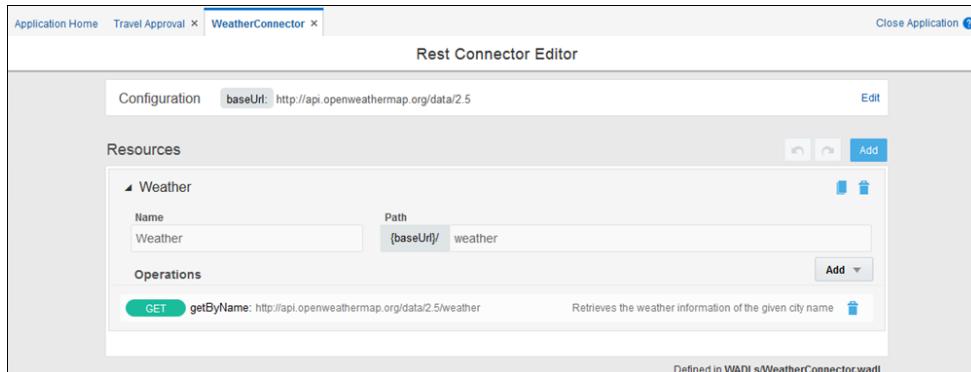
- In the **Security Type** field, select **APP Id - Basic Authentication** to apply basic authentication to the connector.
- Complete the keystore credential fields that display. To create a new key, select **New Key** and enter a key name, user name, and password.
- Click **Apply** to save the configuration settings.

If you edit the value in a field and the **Apply** button doesn't become active, check the value of the **Basic Authentication** field on the Security tab. You must select or add a new key in the Basic Authentication field, or select **None** if no authentication is needed. The **Apply** button only becomes active after the Basic Authentication field has a value.

Want to learn more about managing credentials? See [Configuring Credentials for Web Services](#).

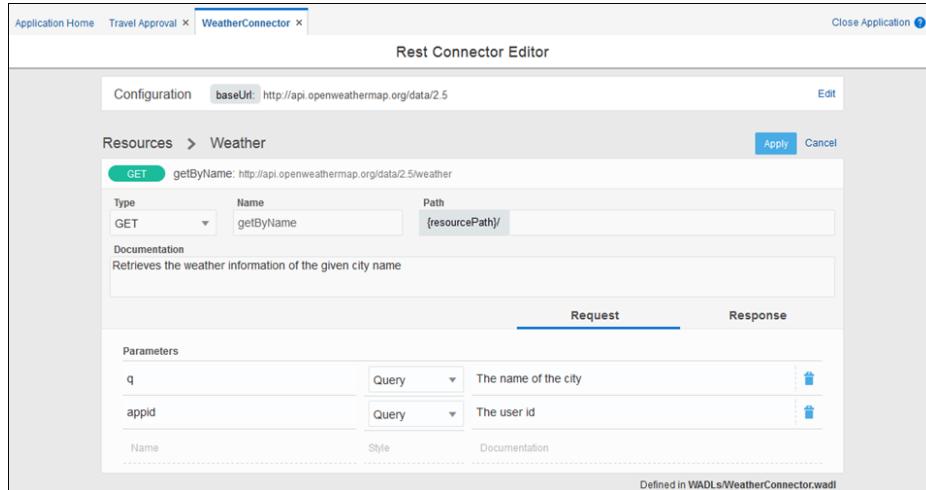
4. Add resources to the connector.

A resource contains one or more operations that control data by performing basic create, read, update, and delete operations (CRUD) on resources using standard HTTP method requests.



- At the top of the Resources section, click **Add**. Click the new resource that was added to the Resources list to expand it.
 - Enter a name for the resource. You'll select this resource name when implementing a service task.
 - In the **Path** field, identify the resource path within the base URL.
 - If needed, duplicate (clone) and delete resources, and undo and redo changes to them, using options in the Resources section. Duplicating a resource can be useful when configuring similar resources.
- Add operations to resources.

- a. In the Operations section, click **Add**.
- b. Select an HTTP method to add (GET, PUT, POST, PATCH, DELETE, or HEAD). The new operation is added to the Operations list, along with its name and path.



6. Specify request and response parameters for operations.
 - a. In the Operations list, select an operation to expand it. If needed, make changes to the operation's **Type**, **Name**, and hierarchical **Path** fields. Optionally, enter a description in the **Documentation** field.
 - b. Click the **Request** and **Response** tabs to view the operation's parameter and body fields. The HTTP method you selected determines the request and response message combination to complete for the operation. An asterisk displays if one or more required fields on the tab need to be completed.
- Refer to the table below for required and optional parameters for HTTP methods. Information in this table is based on the Methods definition section of the [HTTP/1.1 protocol definition](#).

HTTP Method	Description	Request Message-Body	Response Message-Body
GET	Retrieve information from a resource	Optional	Yes
POST	Create a resource	Yes	Optional
PUT	Completely update an existing resource	Yes	Optional
PATCH	Partially update an existing resource	Yes	Optional
DELETE	Delete a resource	Optional	Optional
HEAD	Identical to a GET except that no message body is returned in the response	Optional	No

- c. In the **Body** field, add or select a business object to store the message data. You can create business objects based on JSON files for REST operation payloads. To create a business object by importing or pasting a JSON sample, click **Create Business Object** +, and complete the fields in the Import Business Object from JSON dialog box, uploading or pasting JSON sample

text, and clicking **Import**. The JSON text is validated, and if invalid, an error displays.

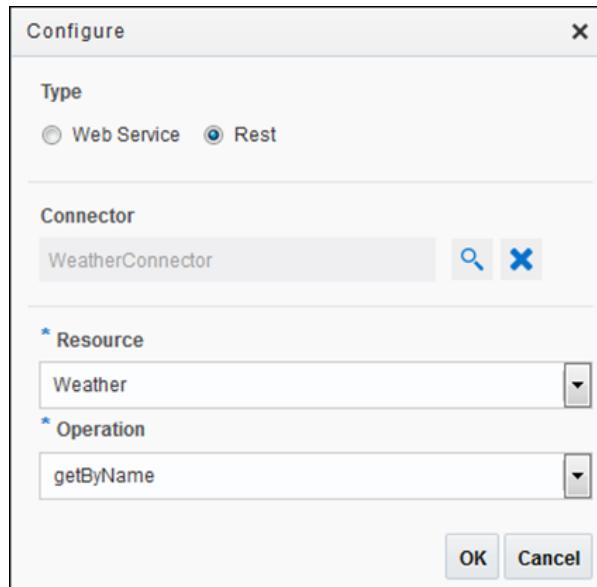
Ensure that the JSON output of the REST connector does not contain elements that start with a number or contain spaces. These elements may throw up a failed mapping error during runtime when converted to XML. Element names are case sensitive, and must start with a letter or underscore.

- d. When you select a business object in the **Body** field, you must also specify the message's media type in the **Media type** field. Oracle Process Cloud Service supports *application/json*.
- e. In the **Parameters** table, configure one or more parameters for the operation. Click an empty row to add a line, and complete name, style, and description fields. Depending on the selected method, supported style options include template, header, body, and query. When you add, edit, or remove a parameter token in the operation's path (for example, {Id}), its corresponding template parameter is automatically created, edited, or deleted.
- f. Click **Apply** to save the operation. Apply is active only when all required information has been entered.

7. Implement the REST connector in one or more service tasks within a process.

In this step, configure one or more service tasks to call the REST service, perform an operation, and store data changes in the business object. See [Invoking ICS Integrations, REST, and Web Services](#). Configure **data association** to pass new values (such as the body message and template) as input and output to the service task.

- a. Add or open a service task in the process. In its properties, implement a service call to a REST service. Select the REST connector, resource, and operation to call.



- b. Using data association, configure input and output for the service task. For example, if the REST connector uses a GET operation to retrieve email content from Eloqua, and a PUT operation to update the email with the response, configure the business object to store the values as they're changed by the service calls.

To delete a REST service connector, click its delete icon on the Integrations page. You can't delete a REST service connector that is being called by a service task.

Working with Web Service Definition Files

From the Definitions view of the Integrations page, you can import a web service definition (WSDL) file, upload a new version of the file, and add a new web service connector file based on a specific port type.

On the Application Home tab, click **Integrations** and then click the **Definitions** view option.

Limitation for Importing a WSDL from URL

To import resources such as WSDLs from an external URL, the URL must be publicly accessible. In other words, the URL can't require any kind of credentials for access. If you need to integrate with such resources, then you should first download them to your computer and proceed to import them from the file.

Importing a WSDL

You can use this import feature to update current versions of WSDL files. However, updating your WSDL files using the import feature is different from the update feature you can find in the Definitions — Details view where you can only update the current WSDL file. When updating WSDL files using the import feature you update the WSDL file and XSD dependencies contained in the ZIP file or imported through a remote URL.

To import a WSDL:

1. Click **Import**  to open the Upload Web Service Definition File dialog box.
2. Select one of the following options:
 - **Upload from file:** Click **Browse** to browse for a WSDL or ZIP file.
 - **Use URL:** Provide the URL for your WSDL or ZIP file.
3. Click **Validate**, and then click **Upload**.

After clicking **Validate** a summary of the files to be added are shown if the validation is successful.

Note:

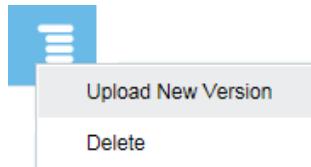
If validation is unsuccessful, instead of showing a summary of the files, the validation errors or warnings are displayed. Validation warnings occur if some of the files that you want to upload override existing ones. In this case you can upload and override existing files. Whenever an unexpected error occurs, the error is logged in the server log and the upload is cancelled. You can't import files that fail validation.

When you accept the import by clicking **Upload**, the operation is committed and the files are uploaded to the application and shown in the Definitions view of the Integrations page.

Uploading a New Version of a WSDL

To upload a new version of a WSDL:

1. Click the name of the WSDL that you want to override.
2. Click the **Options** menu and then select *Upload New Version* to open the Upload New Version dialog.



3. Select one of the following options:
 - **Upload from file:** Click **Browse** to browse for a WSDL or ZIP file.
 - **Use URL:** Provide the URL for your WSDL or Zip file.
4. Click **Validate** and then click **Upload**.

After clicking **Validate** a summary of the files to be added are shown if the validation is successful.

Adding a Web Service Connector

To add a new web service connector based on a specific port type:

1. Click **Add** to open the Add Service Connector dialog. Enter a name and select the port type. You can also select a callback port type if required.
2. Expand **Advanced** and provide the following information, and then click **OK** to add the new web service connection.
 - **Read Time Out:** Specify the time out for reading the WSDL file in milliseconds.
 - **Connection Time Out:** Specify the time out for connecting to the web service in milliseconds.
 - **Security:** Select *None*, *APP Id – Basic Auth*, *APP Id – Username Token*, or *APP Id – Username Token With Message Protection* from the drop-down list.
See [Applying Message Security to Integrations](#).
 - **Certificate:** If the Security is set to *APP Id — Username Token With Message Protection*, select **New Certificate Alias** from the drop-down list and click **Add Certificate** to upload a certificate.
See [Managing Security Certificates during Design Time](#).
 - **Keystore Credential:** If Security isn't set to *None*, select a key from the drop-down list.

You can also select **[New Key]** and type a *Name*, *Username*, and *Password*. For an existing key, the **Username** and **Password** values are automatically populated.

Want to learn more about how to add credentials? See [Configuring Credentials for Web Services](#).

Preparing Local WSDL Files Containing Remote References

If a WSDL file has dependencies on remote schema files, you can download the files and set up local references. Note that this step is needed if creating from a local file only. If creating from a remote URL, Process handles remote references for you.

You must follow the same steps for a process exposed as a web service or for a web service created outside of Process.

To prepare a WSDL file with remote references for use in a web service connection in Process:

1. Create a root directory with two subdirectories named `WSDLs` and `Schemas`.

You can create additional subdirectories of `WSDLs` and `Schemas`, as long as all files under `WSDLs` are WSDL files and all files under `Schemas` are schema files.

2. If the WSDL file is only accessible using a URL, download it and copy it to the `WSDLs` directory.
3. If the WSDL file has an extension other than `.wsdl`, such as `.xml`, rename it and give it a `.wsdl` extension.
4. Open the WSDL file in a text editor.
5. Search the WSDL file for URLs that reference schema (`.xsd`) files.

For example, a schema file reference might look like this: `schemaLocation="http://www.example.com/ExampleService/ExampleSchema.xsd"`

6. Download each schema file and copy it into the `Schemas` directory or a subdirectory of `Schemas`.
7. In the WSDL file, change each schema file reference to refer to the `Schemas` directory.

For example, an edited schema file reference might look like this:

`schemaLocation=" /Schemas/ExampleService/ExampleSchema.xsd"`

8. Save and close the WSDL file.
9. Zip the root directory that contains the `WSDLs` and `Schemas` directories.

Make sure the zipping process doesn't create any extra files in the `.zip` file.

The `.zip` file must contain only `WSDLs` and `Schemas` directories and subdirectories that in turn contain only WSDL and schema files.

Use the resulting `.zip` file to create a connection to the web service. See [Creating a Web Service Connector](#).

Creating a Web Service Connector

When creating a connection to a web service using Process, you specify a WSDL file that is stored locally on your machine or available online. For a secure web service, you can ensure that only users with the proper credentials can access it.

If you're connecting to a secure web service outside of Process, you must obtain the credentials according to the instructions for that web service.

When you deploy a process beginning with a message start event, the process is exposed as a web service. The WSDL file URL is displayed when you select **Web Services** from the Actions menu on the Manage Deployed Applications page. You can

then connect to the process just as you would to any other web service. See [Communicating Between Processes](#) and [Administrator Basics](#).

To configure credentials for a process exposed as a web service, you must have administrator privileges in Process. You can create a security key in two ways:

- When you create a web service connector in Process
- When you select **Configure** in the Home page

Want to learn more about configuring credentials in Workspace? See [Configuring Credentials for Web Services](#)

1. Create a web service connector in any one of the following ways:

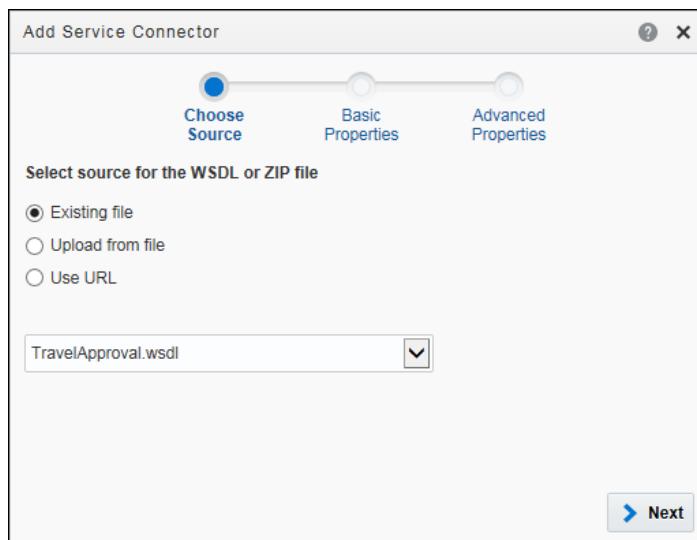
- From the Application Home tab
- From the properties of a service task

To add a connector from the Application Home tab:

- a. On the Application Home tab, click **Integrations** and select the **Integrations** view.



- b. Click **New Integration** +, click **External**, and then select **SOAP**.



To add a connection from the Properties dialog box of a service task:

- a. Add or open a service task in the process. In its properties, implement a service call to a web service.
 - b. Click **Add** +.
2. Select the source for the WSDL or Zip file, then click **Next**:
 - **Existing File**: Select an existing definition from the drop-down list.
 - **Upload from file**: Import a WSDL or ZIP file located on your local machine. A ZIP file can only contain WSDL and XSD files. The WSDL file must have valid references.
 - **Use URL**: Enter a URL address.

The WSDL is retrieved along with other depending files. These files are stored locally inside the application.

 **Note:**

When you click **Next**, the system automatically validates your selected files before importing.

3. In the Basic Properties dialog box, provide the following information:
 - **Name:** Enter a name for your service connector or use the default. Don't use spaces or special characters such as &%\$#/?.
 - **Port Type:** Select a port type from the drop-down list. A WSDL file exposes one or more port types for the service it defines.
 - **Callback Port Type:** Select a callback port type from the drop-down list.
4. Click **Create** to create your web service connector now, or **Next** to open the Advanced Properties page.
5. If you clicked **Next**, complete the Advanced Properties settings, and click **Create**:
 - **Read Time Out:** Specify the time out for reading the WSDL file in milliseconds.
 - **Connection Time Out:** Specify the time out for connecting to the web service in milliseconds.
 - **Security:** Select *None*, *APP Id – Basic Auth*, *APP Id – Username Token*, or *APP Id — Username Token With Message Protection* from the drop-down list.
 - **Certificate:** If the Security is set to *APP Id — Username Token With Message Protection*, select **New Certificate Alias** from the drop-down list and click **Add Certificate** to upload a certificate.
Want to learn more about how to add certificates? See [Managing Security Certificates during Design Time](#) and [Applying Message Security to Integrations](#).
 - **Keystore Credential:** If Security isn't set to *None*, select a key from the drop-down list.
You can also select **[New Key]** and type a *name*, *username*, and *password*.
For an existing key, the **Username** and **Password** values are automatically populated.

After you create a web service connector, you can only change the port types and the advanced settings. To change other settings, you must delete and recreate the web service connection.

You can't create business objects based on in-line types defined in a WSDL file. However, read-only business objects are automatically created from the WSDL file when you create a web service connector. You can create data objects based on these read-only business objects. Want to learn more about business objects and data objects? See [Managing Application Data](#).

Applying Message Security to Integrations

To protect message exchange for your integrations , especially when interacting with external cloud services leveraging SOAP, apply message protection. This protection applies request/response message encryption using OWSM message policies to the connection. As part of message security, you apply keystore credentials and security certificates. After you apply message protection, you can manage certificates and credentials in multiple ways in Process.

To configure message security:

1. Add or edit an integration. See [Creating a Web Service Connector](#) and [Configuring ICS Integrations](#).
2. In the Advanced Properties options, select **APP Id - Username Token With Message Protection**.

The window expands to include certificate alias and keystore credential fields.

The screenshot shows the 'Add Service Connector' dialog box. The 'Advanced Properties' tab is selected. The 'Security' section is set to 'APP Id - Username Token With Message Protection'. The 'Keystore Credential' dropdown is set to '[New Key]'. The 'Username' field contains 'User Name' and the 'Password' field contains '*****'. At the bottom, there are 'Back' and 'Create' buttons, with 'Create' being the active button.

3. Complete the credentials fields.

Select a design-time key or select [New Key] and add one, entering a username and password. If needed, administrators can update credentials in runtime, as described in [Configuring Credentials for Web Services](#).

4. In the **Certificate Alias** field, select a design-time certificate or select [New Certificate Alias] and create a new one. To create a new design-time certificate, click **Add Certificate**, enter an alias name, and either upload a file or paste certificate content.
5. Manage certificates and their expiration dates as needed.

You can manage and maintain separate certificates for design time and runtime, applying them as needed during deployment. See [Managing Security Certificates during Design Time](#) and [Managing Security Certificates during Runtime](#).

Invoking ICS Integrations, REST, and Web Services

Using the connectors that you created, your applications can communicate with local and remote applications that are exposed as SOAP, REST, or web services.

If you choose not to display the integration, REST, or web service connector in the elements palette, you can implement the integration, REST connector, or web service in a service task in a process.

To invoke an ICS integration, or a REST or web service:

1. On the Application Home tab, click **Processes**, and then click the name of the process to which to invoke the ICS integration, or the REST or web service.
2. Drag and drop a Service flow element into the process.
3. Select the Service flow element, click **Menu** , and then select **Open Properties**.
4. Select **Service Call** from the **Type** drop-down list.
5. Click **Edit**  to display the Configure dialog.
6. Select *ICS , Web Service or Rest*, then click **Browse** to display a list of ICS integrations, or REST or web service connectors.
7. Select an active ICS integration, or service connector, then click **OK**.
 - *Integration*: Select an active integration, then click **OK**.
 - *REST Connector*: Select a resource and operation, then click **OK**.
 - *Web Service Connector*: Select an operation, then click **OK**.

Embedding Process UI Components in Other Applications

Integrate Process functionality in external applications, such as self-service applications, services, or portals, by embedding Process UI components provided as jQuery widgets. The external application or service consumes the embeddable Process component and renders it on the appropriate mobile, tablet, or web platform.

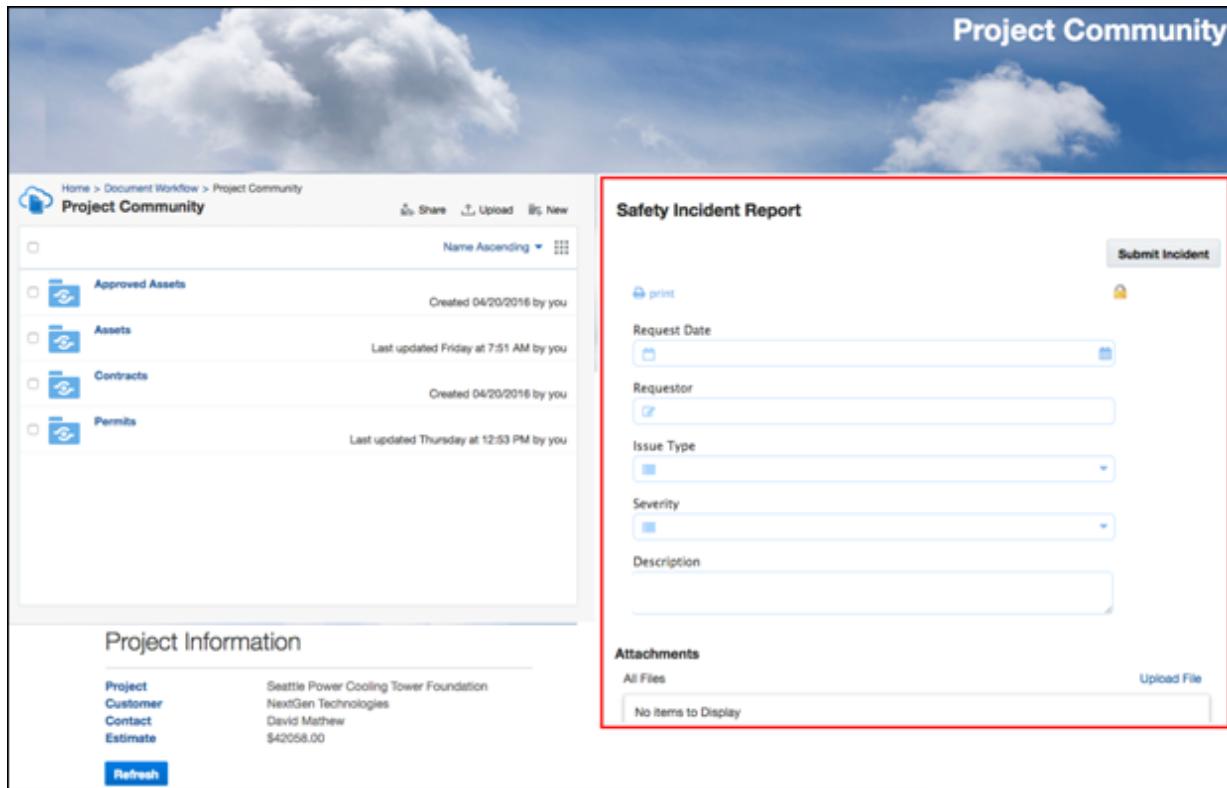
Topics

- [Process UI Components Available for Embedding](#)
- [Before You Embed Process UI Components](#)
- [Downloading Process UI Component Files](#)
- [Experimenting with Process UI Components](#)
- [Consuming the Process UI Component](#)



Process UI Components Available for Embedding

After you embed Process UI components in an external application, your users can then start their tasks within the context of their primary focus rather than in Process runtime. For example, you might configure a Process form that initiates a process by enabling users to submit a safety incident report.



Use jQuery to embed a selected reusable Process UI component in any HTML page. Components related to the following categories are provided.

- **Task**
- **Application:** Display the list of applications and their start form or start form and attachments or documents. Documents display only if Oracle Content and Experience Cloud is configured, and only if documents are enabled for the application..
- **Charts**
- **Miscellaneous**

Embeddable Process UI components expose several options and events that the consuming application can use. You can customize the components using these options and get access to the internal data using public methods.

Whenever a significant user action occurs, the components trigger events. For example, when a user submits a form, an event is fired along with the instance data. You can listen to the events, and perform actions on the consuming application. See <http://learn.jquery.com/events/introduction-to-custom-events/>.

Before You Embed Process UI Components

Review these requirements and tips before you embed Process UI components in external applications.

- Embedding components requires knowledge of JavaScript, HTML, jQuery, and Require JS.
- An open endpoint to Process UI components is provided, and the cookbook is hosted directly on it. The open endpoint gives you several options:
 - Download and use the code in the zip file
 - Point directly to the Oracle Process Cloud Service server to load JS/CSS resources remotely without the need to include the resource inside your application
 - Use the HTML Inline Frame Element <iframe> to embed the component pages
- Embeddable components are automatically protected by the same security model as runtime. To implement OAuth security for a component, use an OAuth token to access the UI. See [Using OAuth with REST APIs](#).
- Downloading files, including the entire library of embeddable Process UI component files and dependencies, are provided. See [Downloading Process UI Component Files](#).
- Newer versions of the component files may be available with each release, although you can download previous versions. Note that any customizations you make to previous versions must be made again to a newer version.
- To view and experiment with the UI component capabilities, use the cookbook. See [Experimenting with Process UI Components](#).
- To change the look and feel of either a single component or the entire UI, override the cascading style sheet (CSS). See [Consuming the Process UI Component](#).
- Embeddable components and REST APIs support cross-origin resource sharing (CORS).

Downloading Process UI Component Files

Before you can experiment with the Process UI components, you need access to the entire code base of the embeddable components, including the Process library, cookbook sample code, and dependency library. The library contains both the debug (complete) version and the minified version for JS and CSS files. You can download and host the component files. Alternatively, you can use an HTML Inline Frame Element (<iframe>) to embed the components.

To download and host the component files:

1. Access the open components endpoint on your Oracle Process Cloud Service server using the following format:
host : port /bpm / components
2. Click **Resources**.
3. Download the files listed in the **Download** section.

Be sure to download and update component files after releases as needed. The Process library file identifies the component feature's version number. To ensure that you're using the latest embedded Process UI component files available in future releases, compare your current version number with the version number listed on the Resources page, and download the newer version if needed.

As an alternative to hosting the components, you can use them in an <iframe> element instead. For information, go to the **Resources** page, scroll to the **Server Access** section, and expand the **HTML** section.

Experimenting with Process UI Components

Use the cookbook to try out Process UI component functionality, and view supported methods, events, and options. Note that the cookbook is provided for demonstration purposes only. You can't use it to consume the UI components.

To experiment with Process UI components:

1. Access the cookbook at the following endpoint:

host:port/bpm/components

The Home page opens. It displays settings and information that apply to all the embeddable Process UI components.

2. On the Home page, complete the **Server Setup** fields to identify your Oracle Process Cloud Service server.
 - You must configure these fields if you want to experiment with using the Process UI components.
 - Select the **testMode** option if you want to get and test production data. This option is equivalent to the runtime test mode.
3. Click **Apply**.

The screenshot shows the Oracle Process Cloud Service Embeddable Process UI Components interface. At the top, there's a navigation bar with links for Home, Resources, DevTool, and Cookbook. The main content area has a sidebar on the left with sections for Prerequisites, Process Cloud Service Connection, and Configuration. The Process Cloud Service Connection section contains a list of prerequisites: Basic understanding of JavaScript and HTML, Basic understanding of jQuery and RequireJS, and Familiarity with jQuery widgets. The Configuration section contains fields for Server Setup: serverURL (http://host-address:port-number), username (jsmith), password (redacted), authInfo (Basic d2VibG9naWM6d2VibG9naWM), and testMode (radio button). Below these fields is an 'Information' section with the following text:

- The Components UI is based on Oracle JET 2.0 and Oracle Process Cloud Service REST API 3.0.
- Component options are case sensitive. Be sure to pass them correctly.
- Events fired by components are on the root element where the component code is injected.
- HTML data elements have either a unique ID or style class defined. Use the Cascading Style Sheets (CSS) selector to change the look and feel of an element.

4. While you're on the Home page, expand and review the information:
 - In the **Process Cloud Service Connection** section. It describes how to connect to the Process server and authenticate the user using a Basic, JWT, or OAuth access token.
 - In the **Configuration** section. It describes how to consume the embeddable components in the external application's main.js file.
- You'll need this information when you're ready to connect and consume the components.
5. Click **Cookbook**.
 6. Use the links in the left pane to select a UI component (**Detail**, **Start Application**, **Start Form**, **Attachments**, **Comments**, or **Conversation**).
 7. Expand the **Documentation** section and review its details about configuring the selected component. The Documentation section displays the following information:
 - Require**: Describes the Require JS module name and any other settings required to consume the component.
 - Usage**: Provides sample code for consuming the component.

- **Options:** Provides a table listing ways to alter the component's behavior. Some of the options are required to consume the component. Expand the **Component Setup** section to try out the options.
 - **Methods:** Lists methods to access various data in the component. For example, use the applist component's `getStartformList` method to get a list of available forms. This information is useful for consuming the Start Form component independently. Expand the **Data** section to view event and method call results.
 - **Events:** Lists the user actions on the component that fire events that the consuming application can listen to and use in the external application's logic. Expand the **Data** section to view event and method call results.
8. Expand the **Demo Code** section to review sample JS and HTML code for the selected component, along with the component options currently being used to load the component. These options change when you change Component Setup values.
 9. Expand the **Component Setup** section, select or enter options, and click **Apply** to see how your changes affect the component. For example, try renaming or hiding buttons.

The screenshot shows the Oracle Process Cloud Service Embeddable Process UI Components interface. The left sidebar has a navigation menu with items: Task, Detail, Application, Start Application, Start Form (which is selected and highlighted with a dashed border), and Miscellaneous. Under Miscellaneous, there are links for Attachments, Comments, and Conversation. The main content area is titled "Start Form" and contains sections for Documentation, Demo Code, Component Setup, and Data. Below this, a form titled "Start the process PD01 and PD02(1.0)" is displayed. The form has fields for mode (with a dropdown icon), appName, appVersion, entityId, and recordKey, each with a small edit icon. To the right of the form are three buttons: Submit, Save, and Discard. At the top of the main content area, there are links for Home, Resources, DevTool, and Cookbook.

Consuming the Process UI Component

Use the configuration, documentation, and demo code provided in the cookbook to guide you in integrating the component:

1. Consume the downloaded library's js/libs and css/libs inside your external application.

Alternatively, you can directly point the 'pcs' path in your requirejs file to the remote Oracle Process Cloud Service server. For example:

```
'pcs' : 'PCS_SERVER_URL/bpm/components/js/libs/pcs/v1.2/min'
```

This command means you don't have to include the Process code in your application, and can get it remotely instead.

2. Create the Process connection (\$.pcsConnection) in the app main.js file and configure it to identify the server URL and authentication. Go to the Home page and expand the **Connection** section for details.

The authentication information is provided through the authInfo property, which can contain the Basic, JWT, or OAuth access token. The app main.js file is responsible for maintaining a valid token.

Basic authentication token can be provided by:

```
var tok = username + ':' + password;  
$.pcsConnection.authInfo = 'Basic ' + btoa(tok);
```

Other access tokens (JWT or OAuth) can be provided by:

```
$.pcsConnection.authInfo = 'Bearer {token string}'
```

See [Obtaining a Client Access Token](#).

3. Configure component use in the app main.js file. For details, go to the Home page and expand the **Configuration** section.
4. Configure the selected component, as described on the Cookbook UI's component pages. For example, see Require, Usage, Options, Methods, and Events fields.
5. Optionally change a component's look and feel, or that of the entire UI, by overriding the cascading style sheet (CSS) files.

Each UI component has an ID and a class. For example, to change the application list links from circles to squares, override `.pcs-applist-process-holder-smaller` css class.

- a. To change a component's css, expand the **Demo Code** and **HTML** sections on the component page and determine the CSS file. You can copy, modify, and override the CSS file listed.
- b. Use CSS selectors to access HTML elements and modify the look and feel after the UI component has rendered.

Troubleshooting Application Development

Here are common issues you might encounter using Process and possible solutions.

For additional information that might be helpful in troubleshooting, see Known Issues in Composer in *Known Issues for Oracle Process Cloud Service*.

Topics

- [Troubleshooting Application Import](#)
- [Troubleshooting Business Processes](#)
- [Troubleshooting Basic Form Rules](#)
- [Troubleshooting Data Association](#)
- [Troubleshooting Business Rules](#)
- [Troubleshooting Application Playing](#)

Troubleshooting Application Import

The following troubleshooting tips pertain to importing applications.

When I import an application, I receive an error stating that an application with the same name already exists

Application names must be unique, even if they have different owners or are in different spaces. For example, if two users try to upload one of the QuickStart Apps, the second user will be unsuccessful.

To solve the problem, rename the application:

1. Unzip the .exp file.
2. Rename the resulting root directory.
3. Zip the renamed root directory to create a new .zip file.
4. Change the .zip extension to .exp.

When I import a renamed application, I receive an error stating that the project Info file is invalid

Some unzipping commands put the root directory under a new directory with the same name. Then when you zip the new directory thinking it's the root directory, the file structure is not recognized. This structure is incorrect:

```
Application Name (added)
  Application Name (root)
    SOA
      (other subdirectories)
```

To solve the problem, make sure the root directory that gets zipped contains only one subdirectory, named SOA. This structure is correct:

```
Application Name (root)
  SOA
    (other subdirectories)
```

Troubleshooting Business Processes

The following troubleshooting tips pertain to creating and editing business processes.

The Process Editor is a blank screen

To solve the problem, go to <http://helpx.adobe.com/flash-player.html> and follow the instructions to download the latest version of the Flash player. This player is free.

I'm receiving a validation error stating that the condition in a sequence flow is empty

You created a gateway element, created a data object for it, and connected its outbound paths, but you didn't implement the non-default paths.

To solve the problem, perform these steps for each non-default path:

1. Click the path, click the edit icon on the path, and click **Implementation**.
2. Type a name for the path.
3. Click **Edit** next to the Condition field.

The Expression Editor appears.

4. Type an expression that tests whether the gateway data object has one of the outcome values.

For example, if the gateway handles the outcome of an approval task, the name of the gateway data object is approvalOutcome, and the APPROVE outcome is the default path, type `approvalOutcome == "REJECT"`. Note the double equal sign.

5. Click **OK** to close the Expression Editor and then close the implementation pane.

You don't need to implement the default gateway path.

I created two business objects based on two schema definitions that have unique element names but the same namespace. I am receiving exception errors when I use these business objects in my process.

Schema type definitions on the same `targetNamespace` from two different locations can't be resolved properly. Any attempt to use these schemas such as, when creating a business object or as part of a WebService definition, causes errors throughout the application. Therefore, you should either provide another namespace or define those schema types in the same file.

Troubleshooting Basic Form Rules

The following troubleshooting tips pertain to creating basic form rules.

A basic form rule displays a field under a certain condition, but when the condition is no longer true, the field doesn't disappear

For example, suppose you create the following basic form rule:

```
if (Text1.value == 'Test') {  
    Text2.visible = true;  
}
```

If you type `Test` in the `Text1` field, the `Text2` field appears, but then when you type something else in the `Text1` field, you expect the `Text2` field to disappear, but it remains visible.

To solve the problem, you must explicitly reverse the condition:

```
if (Text1.value == 'Test') {  
    Text2.visible = true;  
}  
else {  
    Text2.visible = false;  
}
```

Similarly, suppose you create the following basic form rule:

```
if (Text1.value == 'Test') {  
    Text2.visible = true;  
}  
else {  
    Text3.visible = true;  
}
```

You might expect `Text2` and not `Text3` to be visible if `Test` is in the `Text1` field, and the reverse if anything else is in the `Text1` field, but this isn't what happens. You type `Test` and `Text2` appears, you type something else and `Text3` appears, and neither disappears.

To solve the problem, you must explicitly reverse both conditions:

```
if (Text1.value == 'Test') {  
    Text2.visible = true;  
    Text3.visible = false;  
}  
else {  
    Text2.visible = false;  
    Text3.visible = true;  
}
```

Troubleshooting Data Association

The following troubleshooting tips pertain to performing data association.

When I click Apply in the data association editor, I receive an error that validation of associations failed

The output data type from the previous process flow element doesn't match the input data type that the current process flow element expects.

To solve the problem, first make sure you dragged and dropped the correct data object. If the data object is correct, add a function that converts the data type:

1. In the data association editor, click the Launch Expression Builder icon next to the input field.

The Expression Editor opens.

2. Surround the data object name with a function named for the type that the current process flow element expects.

For example, if the field name is `inputDataObject`, and the type must be string, type `string(inputDataObject)` in the Expression Editor.

3. Click **OK** to close the Expression Editor and then click **Apply**.

When I click the Validate Application icon, I receive an error stating that data association isn't valid because double or float can't be assigned to int

A field with a data type of int can only contain integers, or whole numbers. A field of type double or float can contain decimal numbers.

In data association, if the input type is int, the output type can be any numeric data type, such as int, double, or float. However, if the output type is int, the input type must also be int, or an error occurs.

You can solve the problem in one of these ways:

- Change the data type of the input field to int.
- Change the data type of the output field to double or float.
- Use an expression to convert the input field type to int.

To convert the input field type:

1. In the data association editor, click the Launch Expression Builder icon next to the input field.

The Expression Editor opens.

2. Surround the field name with the `round()` and `int()` functions.

For example, if the field name is `loanAppDataObject.form.income`, change it to `int(round(loanAppDataObject.form.income))`.

3. Click **Validate** to verify the expression and then click **OK**.

The Expression Editor closes.

For example, if you're performing data association for a Decision and the inputs are from a form, you can solve the problem in one of these ways:

- In the form editor, use a Quantity field, which has a data type of int.
- Recreate the Decision and make sure the Fact Type is double or float.
- Use an expression to convert the input field type to int as described previously.

Troubleshooting Business Rules

The following troubleshooting tips pertain to creating and editing business rules.

I am receiving a validation error stating that *decision.out* isn't an input

When you created the Decision, you did both these things:

- You specified an output fact that was not also an input fact.
- You used an action other than create, such as modify, on this output fact.

You can solve the problem in either of these ways:

- Recreate the Decision so that the same fact is both an input fact and an output fact.
- Use a create action on the output fact. This makes creating an input fact to match the output fact unnecessary.

Troubleshooting Application Playing

The following troubleshooting tips pertain to using the application player.

I'm receiving an error stating that there was a problem deploying the application and the application player can't be initialized.

The instructions to enable the player were followed, but the credentials entered were incorrect. Click **Clear** and try again. See [Enabling the Application Player](#).

I'm still receiving the message that *Credentials aren't configured* after the administrator entered them.

If your application was opened before the administrator entered the credentials, you must close and reopen your application to enable the application player for that application.

Part III

Administrator Tasks

Topics

- Administrator Basics
- Managing the Runtime Environment
- Managing the Design-Time Environment

15

Administrator Basics

Users assigned an administrator role can perform a variety of configuration, management, and monitoring tasks in Design Time and Runtime.

Topics

- [About Administrator Privileges](#)
- [Configuring Federated SSO and Authentication](#)
- [Using OAuth with REST APIs](#)

About Administrator Privileges

If you're assigned to the Administrator role in Process, you have access to the Administration pages in design time and runtime.

Administrator privileges are granted by assigning the Administrators group to an Process user. Want to learn more about granting administrator privileges to a user? See [About Process Roles and Users](#).

Runtime administration: Click your user name in the top-right corner and select **Administration** to assign roles to users, configure notification logs, configure Oracle Content and Experience Cloud, and much more.

Design-time administration: In the global toolbar, click:

- **Administration** to manage spaces, applications, and the application player.
- **Management** to activate applications to your test and production environments.

Configuring Federated SSO and Authentication

Federated single sign-on (SSO) delegates user authentication to a trusted external entity, rather than having the Shared Identity Management (SIM) directly authenticate the user. Your on-premise identity system, such as Oracle Identity Federation (OIF) or Active Directory Federation Services (ADFS), challenges the user for sign-in credentials, and then asserts the user's identity to the Oracle Cloud SIM system, using Security Assertion Markup Language (SAML) as the protocol.

After SIM has accepted the user's identity and established an authenticated session, the rest of the application authorization and access management proceeds as usual. As long as the user identity asserted by the on-premise identity provider maps to the expected user identity in SIM, then whatever groups, roles, and attributes the user has in the SIM directory are available in Oracle Process Cloud Service at runtime.

Creating the User Accounts

In order for Federated Single Sign-On (SSO) to work, these conditions must exist:

- Accounts for each user must exist in both the Oracle Cloud / Shared Identity Management (SIM) environment and the on-premise identity provider
- The user's email address used in both accounts must be the same

To create the user accounts in the Oracle Cloud / SIM environment, sign in to the My Services application and go to the Users page. On the Users tab, add accounts and assign roles to users. Want to learn more about creating these user accounts? See Adding Users and Assigning Roles in *Getting Started with Oracle Cloud*.

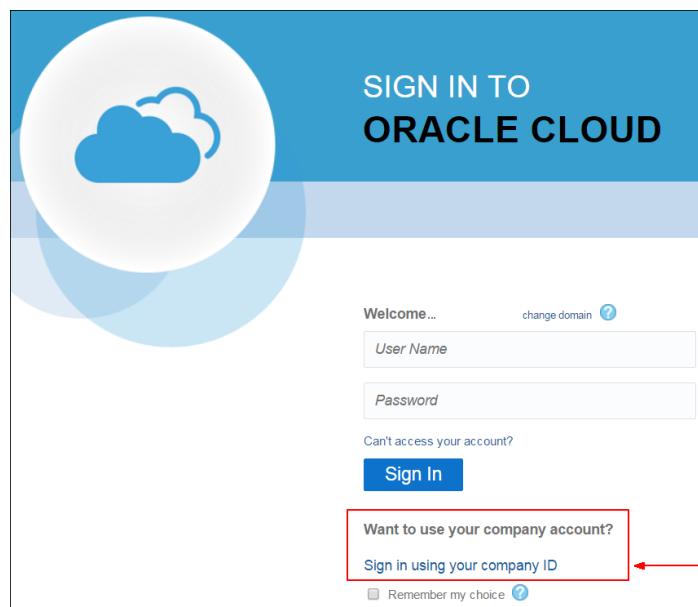
For each account, make sure you enter the email for the user's manager in the Manager Email field. In Oracle Process Cloud Service, many processes are defined as a sequence of tasks or events requiring a series of management approvals. As items in the process are approved, the next task gets assigned based on the management levels defined in the user's account in My Services.

You must also confirm that accounts for all users exist in the on-premise identity provider. Add new accounts if necessary.

Configuring the Providers and Enabling Single Sign-on

To configure federated SSO:

1. Configure Oracle Cloud as a service provider.
 - a. Sign in to the My Services application, click **Users**, click the **SSO Configuration** tab, and then click **Configure SSO**.
 - b. Enter the metadata for your identity provider. You can either import the metadata or enter it manually.
See Configuring Oracle Cloud as the Service Provider in *Administering Oracle Cloud Identity Management*.
2. Configure your identity provider.
 - a. Scroll to the **Identity Provider** section of the page to obtain what you need to configure your identity provider.
 - b. Export the metadata or copy and paste the required information. You configure your identity provider by using its configuration interface. The configuration steps are specific to each identity provider.
3. Return to the **SSO Configuration** page in My Services.
4. Test your federated single sign-on configuration. This process will identify issues that you must fix before you can enable single sign-on.
 - a. Scroll to the **Test your SSO** section of the page, click **Test**, and then click **Start SSO**. You're redirected to the on-premise identity provider.
 - b. Sign in using your administrator user name and password.If the test is unsuccessful, view the test results to try to determine the cause. Make changes to your configuration and repeat the test until it's successful.
5. Scroll to the **Enable SSO** section of the page and click **Enable SSO**.
You must specifically enable SSO before using it. When SSO is enabled and a user enters the URL for Oracle Process Cloud Service, the Sign In page includes a **Sign in using your company ID** link.



When users enter their valid credentials, they will be authenticated through the identity provider and redirected to the home page for Oracle Process Cloud Service Workspace or Composer.

For more details about these configuration steps and how to configure federated SSO, refer to Managing Single Sign-On in *Getting Started with Oracle Cloud*.

Also, the [Configure an Identity Provider with Oracle Cloud - Tutorial Series](#) guides you through the configuration steps for different identity providers.

Using OAuth with REST APIs

The REST APIs for Oracle Process Cloud Service support basic auth, JSON Web Token (JWT), and OAuth for authentication. OAuth 2.0 is an authorization framework that enables an application or a service to obtain limited access to a protected HTTP resource. In OAuth, the applications are called clients; they access protected resources by presenting an access token to the HTTP resource.

Oracle Process Cloud Service accepts OAuth tokens as an alternative to basic auth. As an administrator, you configure OAuth resources and clients. Developers can use the client information you provide to obtain access tokens for the clients.

Note:

Shared Identity Management (SIM) users can use either basic auth or OAuth to access the REST APIs for Oracle Process Cloud Service. Federated single sign-on (SSO) users must use an OAuth access token to access the REST APIs.

Topics

- [Configuring OAuth Resources and Clients](#)
- [Obtaining a Client Access Token](#)

Configuring OAuth Resources and Clients

As the administrator, you're responsible for configuring and managing OAuth resources and OAuth clients in Oracle Cloud. You use the OAuth Administration page in the My Services application to register new OAuth resources and clients, grant and revoke client access to Oracle Cloud APIs, and manage the settings of the resources and clients.

A **resource** is a protected service in Oracle Cloud. When you register a new resource, you define some parameters and these parameters are used in authorizing the client request to those services.

To register an OAuth client for an Oracle Process Cloud Service instance:

1. Sign in to the My Services application. Be sure to sign in to the correct identity domain.
2. Click **Users**.
3. Click the **OAuth Administration** tab.
4. Register your Oracle Process Cloud Service instance as a resource by entering its base URL.
5. Register an OAuth client and associate your newly created resource.

Registering your Oracle Process Cloud Service as a resource provides you with two important values for secure access:

- Client ID
- Client secret

Developers can use this information to obtain an access token for the client.

See Managing OAuth Resources and Clients in *Administering Oracle Cloud Identity Management*.

Obtaining a Client Access Token

The client ID and client secret of the client application are base64 encoded and sent in the header. For example, the authorization header has a value of `base64encoded(client_id:client_secret)`. This value is sent to obtain a client token.

To obtain a client access token:

1. Obtain a client assertion. You can obtain a client assertion in one of the following ways:
 - By providing the client credentials
 - By providing another self-issued JWT assertion
 - By providing another assertion (from an IDM OAuth-generated client assertion or any other third-party JWT assertion)

You access the token endpoint of the OAuth server by passing `client_id:client_secret` as a basic authorization header. The administrator for your Oracle Process Cloud Service can provide you with the client ID and secret for the service instance.

For example, the following cURL command obtains a client assertion by providing the client credentials. Note that the grant type is `client_credentials`.

```
curl -i -H 'X-USER-IDENTITY-DOMAIN-NAME: OAuthTestTenant125'
-u <client_id>:<client_secret>
-H 'Content-Type: application/x-www-form-urlencoded; charset=UTF-8'
--request POST http://<identity-domain>.<data-center>.oraclecloud.com/oam/oauth2/
tokens
-d 'grant_type=client_credentials'
```

See Managing OAuth Resources and Clients in *Administering Oracle Cloud Identity Management*.

2. Obtain an access token.

You can obtain an access token by using different scenarios in the password flow. These scenarios include using the user credentials with either the client credentials or a client assertion.

For example, the following cURL command obtains an access token by passing the user credentials and a client assertion:

```
curl -i -H 'X-USER-IDENTITY-DOMAIN-NAME: OAuthTestTenant125'
-u <client_id>:<client_secret>
-H 'Content-Type: application/x-www-form-urlencoded; charset=UTF-8'
--request POST http://<identity-domain>.<data-center>.oraclecloud.com/oam/oauth2/
tokens
-d 'grant_type=password
&username=name52
&password=test11
&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-
bearer
&client_assertion=eyJhbGciOiJSUzIlNiIsInR5cCI6IkpxVCIsIng1dCI6Ild3cmVwdTJkYXNhSX
HUilBbFZwSGtVQjZK
ZyIsImtpZCI6Ik9BdXRoGVVzdFRlrbmFudDEyNS5jZXJ0In0.eyJvcmFjbGUub2F1dGgudGtfY29udGV4d
CI6ImNsawWvudF9hc3Nlc
nRpB24iLCJleHAiOjE0MjYwMzI4MzgwMDAsInN1YiI6IjMwM2EyNDkyLWQ2NGYtNGUwNC1iNzhmLWI0Mz
MwMDQ3MzEyYiIsImlzcy
I6Ik9BdXRoGVVzdFRlrbmFudDEyNSIsInBybiI6IjMwM2EyNDkyLWQ2NGYtNGUwNC1iNzhmLWI0MzMwMDQ
3MzEyYiIsImp0aSI6IjY
yNzZhYTI0LTUxNjQtNGEwZC1iyzQxLTlmMzVjMGU1ZjgxZiIsIm9yYWNsZS5vYXV0aC5zdmNfcF9uIjoi
T0F1dGhUXN0VGVuYW50
MTI1U2VydmljZVByb2ZpbGUIJCJpYXQiOjE0MjU0MjgwMzgwMDAsIm9yYWNsZS5vYXV0aC5pZF9kX2lkI
joiMTM0NjM2NzUxMzgzM
DI1NjYiLCJ1c2VyLnRlbmFudC5uYW1IjoiT0F1dGhUXN0VGVuYW50MTI1Iiwib3JhY2xlLm9hdXRoLn
Bybi5pZF90eXB1IjoiQ2
xpZW50SUQifQ.OCHS9FhKJExpIg3IvE6qWdTz3tRY449LzoBAcc3yDoaMbjs4CzxDDuKx6MUBpHmkmVoH
RZSmkrIL0ze151st_kjE
HfNtzwMCIs2re_JcSfGkvnzv0aCv1r_V5dvmmZulhGaOUTu9nkEFzCq-JNa23e0_dEq8jfP7-
Y7H2KGMvuC51HGGQViwlega-4mFu
ZBJlSvzEqDcYIPde0m8gSUF--
IFuiovgGTKCe97-0MF34za6SZ0HJv9p3WesvCS8YV1bcWVwTGEXCZ3qA1mA-IOKvaMZNOxM_D9tT
5KV Cub-i-H6r0uHpkov0CzunffcuL4c0g5ptrFv-abn-JP47eNag
&scope=http://www.example.com UPDATE'
```

See Using REST API Calls for the Password Grant in *Administering Oracle Cloud Identity Management*.

3. Confirm that you can access the REST APIs for Oracle Process Cloud Service by using the access token you just obtained.

Managing the Runtime Environment

As an administrator, you manage, monitor, and configure the runtime environment.

Topics

- [Task Overview for Administering the Runtime Environment](#)
- [Assigning and Managing Roles](#)
- [Monitoring and Adjusting Processes](#)
- [Creating Analytics with Oracle Business Intelligence Cloud Service](#)
- [Configuring Application Settings](#)
- [Troubleshooting the Runtime Administration](#)

Task Overview for Administering the Runtime Environment

As an administrator, you are responsible for tracking process instances, monitoring dashboards, and configuring application settings.

Your Responsibilities	Description
Alerts	View alerts for suspended, recoverable, and alerted instances on the Home page, and navigate instance details.
Instances	Track running, inactive, and problematic process instances.
Dashboards	Monitor key process metrics such as workload, cycle time, running process instances, and closed instances.
Administration	Assign users to roles, configure Oracle Content and Experience Cloud, configure and manage notification logs, and much more.

Assigning and Managing Roles

To activate an application to end users, locate the application's roles and assign users, groups, or roles to them. Roles assigned to end users define their permissions, such as task assignment and whether they can start an application.

To assign or manage roles:

1. Go to the Home page, and click **Configure**.
2. Click **Manage Roles**.

The Manage Roles page displays. Below **Manage Roles**, all roles defined in swimlanes for all applications are listed. In addition, several special roles (Process Owner, Process Reviewer, and Analytics Viewer) are automatically created upon application activation and available for assignment. Each role includes the application name followed by its role, and lists its members below. For example, Application1.Process Owner refers to the Process Owner role in Application1. If

needed, enter an application's name in the **Search** field to limit the display to that application's roles.

If needed, add or delete roles. You can add and delete roles to a selected application, as described in [Defining Application Roles](#).

3. Select the role to which you want to assign users.
4. In the **Assign Role** section, click **Add Member**. In the Search users, groups, roles window, search for and select a user, group, or role, using one of the search methods listed. You can use an * asterisk as a wildcard character. Select users from the results, and click **OK**. The selected user, group, or role is now listed in the **Assign Role** table.
5. Click **Save** to save the role assignment. If needed, click **Revert** to discard unsaved changes.
6. Optionally, search for and add a member (user, group, or role) to the **Escalation Path** field to whom to escalate the task.

By default, roles get created for each swimlane defined in an application's processes. In addition, several key application roles are defined for you, as described in the following table. Want to learn more about defining roles? See [Defining Application Roles](#)

Role	Description
Process Owner	Users assigned this role can view process activity history, take actions (such as approve or reject), alter process flow, and view form data for applications they own. Process owners typically manage activated business processes and use metric analysis tools such as dashboards to monitor business processes and alter task flow as needed.
Process Reviewer	Users assigned this role can view tracking and process activity history and view or add comments, attachments, or documents for the specified application. Process reviewers cannot take actions on tasks or alter task flow. They aren't process participants, but typically responsible for reporting on current process instance status, such as in a help desk or front office role.
Analytics Viewer	Users assigned this role can create and view business analytics dashboards associated with the specified application. (Analytics viewers can't update tasks.) See Creating and Viewing Business Analytics Dashboards .

Monitoring and Adjusting Processes

Use these options to monitor overall process flow and make adjustments as needed.

Topics

- [Viewing Alerts](#)
- [Tracking Process Instances](#)
- [Altering the Flow of a Process Instance](#)
- [Monitoring Key Metrics in Dashboards](#)
- [Viewing and Resending Email Notifications](#)

Viewing Alerts

If you're an administrator or process owner, your Home page displays alerts with the number of suspended and recoverable instances, and alerted tasks. Click an alert, to view details about the problematic instances or tasks. Select an instance or task and correct the problem.

Tracking Process Instances

Tracking a process instance enables you to identify any problems with the flow or the assignment that might be delaying the process. Depending on your role, you can locate and track the flow of a specific instance, and view the instance history, comments, attachments, details, documents, and conversations.

Users assigned the Process Owner and Process Reviewer roles can track the entire process. Want to learn more about these roles? See [Assigning and Managing Roles](#). To track process instances:

1. Go to the Home page, and click **Track Instances**.
2. Click the instance you want to track.
3. In the Instance Details pane, view the following:
 - Open Activities: displays the currently running activities in the process. At a certain point in the flow, the process might have multiple activities running at the same time.
 - Comments: displays the history of the comments for this process instance.
 - Attachments: displays the attachments for this process instance.
 - History: enables you to view the process flow this instance followed before getting to the current activity. You can view the history as a list, a tree, or a graphic.
 - Details: displays detailed information about the instance such as the priority, the status and the creation date.
 - Documents: displays the documents that are integral to the instance.
 - Conversations: enables you to view the conversations related to the instance.

You can change the process flow of an instance that is currently suspended because of a problem, or move an instance that is running to another activity. You can also modify values that are causing an activity to fail and then retry running the current activity again. See [Altering the Flow of a Process Instance](#).

Altering the Flow of a Process Instance

You can change the process flow of an instance that is currently suspended because of a problem, or move an instance that is running to another activity because of a specific reason. If an activity is failing because of the value of the data objects and instance attributes, then you can modify them and retry running the current activity again.

To alter the flow of a process instance:

1. Go to the Home page, and click **Track Instances**.

Alternatively, you can access suspended instances, recoverable instances, and alerted tasks from the alerts box on the home page. See [Viewing Alerts](#).

2. Click the instance you want to alter.
3. In the Instance Details pane, click **Alter Flow & action**.
 - If the status of the selected instance is *In Progress*, then the button is labeled **Alter Flow & Suspend**.
 - If the status of the selected instance is *Suspended*, then the button is labeled **Alter Flow & Resume**.
4. In the Open Activities table, go to the New Activity column and select an *activity* that redirects the process instance.
5. In the **Comments** field, enter a comment justifying your action or adding information.
6. Optionally, you can change the value of the data objects or instance attributes:
 - a. To view the process instance attributes, click **Show Instance Attributes**.
 - b. To display the data object in a tree, click **Show as Tree** located below the Data Objects list.
 - c. Click the data object whose value you want to modify.
The XML that represents that data object appears to the right of the list of data objects.
 - d. Locate the value of the data object in the XML and edit it.
7. Click either **Resume** or **Suspend**. Depending on the current status of the process instance only one of these buttons appears.

The process instance moves to the selected activity. Suspended instances are resumed, and running instances are suspended.

Monitoring Key Metrics in Dashboards

Use the dashboards to monitor the overall state of your processes and view specific process metrics such as bottleneck processes. You can also create custom graphs to view process data based on the business indicators defined in your business processes.

Dashboards

To monitor dashboards, go to the Home page, click **View Dashboards**, and select a dashboard. Optionally, click **Filter** to select specific processes to view for that dashboard.



These filters enable you to select which processes to display. For some types of dashboards, you can also select the assignees and time period to display.

Available dashboards are:

Dashboard	Description
Health	Monitor the overall state of the processes available to you, and spot any anomalies. Shows the total number of instances that are in progress, suspended, and recoverable for each process. Click Total Open , Active , Recoverable , or Suspended to view the instances for the selected item or double-click a line in the chart to view the instances for that process.
Open	Shows a top level view of the current running instances, and a detailed analysis of the running instances for each process. This analysis includes the number of instances on track, due this week, overdue, suspended, recoverable, created today and closed today. Click any of the rows to view the instances for that process.
Workload	Shows the workload for the top ten processes or bottleneck processes. View the workload for each of the processes by task or by assignee.
Trend	View the workload and cycle trend for each process, or the workload and cycle trend for each of the tasks in those processes.
Closed	View the closed instances for the current day, week, or month, that are successfully completed, aborted, and errored. Also see the average cycle time for the current period, the previous period, and the maximum cycle time for the current period.

Viewing and Resending Email Notifications

You can view the notification logs to monitor the status of notifications sent for tasks and Business Activity Monitoring. You can try to resend the email notifications to all the original recipients or to some of them by using the notification logs if the email notifications fail due to lack of network availability, wrong email addresses, or temporarily unavailable email servers.

To view the notification logs:

1. Go to the Home page, and click **Configure**.

2. Click **Notification Logs.**

The Notifications table displays the following information:

- Process
- Source
- Recipients
- Sent On
- Status
- Actions

3. On the **Notification Logs page you can:**

- Search for a specific notification.
- Sort the notifications by date or by status.
- View bad addresses.

Click the **View Bad Addresses** button. A dialog box with the list of emails with bad email addresses opens.

- Test notifications.
- Click the **Test Notifications** button. A dialog box opens for you to send a notification email as a test.
- Fix notification problems.

Click the drop-down list on the actions column and select one of the following actions:

- Resend to all members
- Resend to specific members
- Delete

Creating and Viewing Business Analytics Dashboards

Use the Business Data Query to plot and view charts and graphs for application metrics and create Business Analytics Dashboards. You can create charts that display business indicator values (metrics specific to a process) and system indicator values (metrics automatically captured).

When designing a process, developers create **business indicators** for data objects whose metrics they want to capture and display as X axis, Y axis, and filter values. You select the business or system indicators to plot them in charts and graphs. You must be assigned the Process Owner role, or be the Administrator role, or the Analytics Viewer role to create and view business analytics dashboards. See [Assigning and Managing Roles](#).

 **Note:**

You can also export your process analytics data (including business indicators data) using the analytics export REST API and then import them into your own business intelligence tool to create your own business analytics dashboards. See Oracle Process Cloud Service REST API.

To create and view business analytics charts and graphs:

1. Go to the Home page, and click **View Dashboards**.
2. Click the **Business Analytics** tab.

Alternatively, if you're creating a query for the first time, click **Get Started**.

3. Click **New Query**  on the Reports pane.
4. Enter a name in the **Untitled** field.

You can also add a description for the new query in the **Add description** field.

5. In the **Data Source Type** field, select a source for the chart or graph.

Typically, this value is left as Process, the default value.

6. In the **Application** field, select the application to report on.

You can select **All Applications** to report across multiple active applications, using system indicators only.

7. In the **X Axis** options, scroll the **Series** and **Group** fields to see available indicators, and select dimension indicators to plot for the chart's X axis.

Business indicators are listed first, followed by standard system indicators. Want to learn more about dimension indicators? See [Adding Dimension Business Indicators](#). You must specify both a series and a group indicator for the X axis.

 **Note:**

If you select the **Date** data type indicator for the X axis, then you can use the **Time Grouping** option to specify the time frame for the business analytics dashboards of the application. For example, in case of a Travel Request Application, if you select **Year**, you can view the number of Travel Requests by the year.

8. In the **Y Axis** fields, scroll the **Measure** field to see available indicators, and select a measure indicator and standard function to plot for the Y axis.

See [Adding Measure Business Indicators](#).

9. Optionally, add one or more filters. Click **Add Filter**. From the fields that display, click the left one to see available indicators, and select an indicator by which to filter. For available system indicators, see [Selecting System Indicators](#).

Want to learn more about attributes? See [Adding Attribute Business Indicators](#).

10. Click **Display Report**.

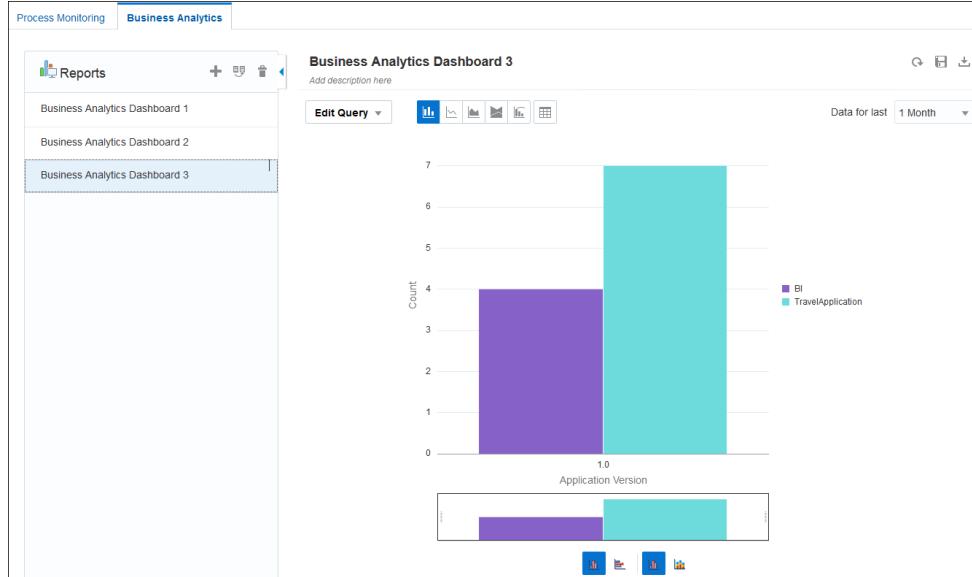
The chart is plotted and displayed on the right side of the page.

- Click **Save** to save the business analytics dashboard that you just created.

After you save the query, you can select and view it on the Reports pane.

 **Note:**

You can use the **Collapse** button to hide the Reports pane when not in use.



- Optionally, change the chart or graph's display.
 - Use the **Data for last** drop down list to select the duration of the activity. By default, the data for last 1 month is selected.
 - Use the icons in the top and bottom to change the chart's type and orientation and stack.
- You can **Delete** or **Copy** the Business Analytics dashboard or **Reset** the fields for **X axis** and **Y Axis** and create a new graph.

14. Optionally, export the data to a file, click **Download CSV**, and open or save the export.csv file.

Selecting System Indicators

You can select the system indicators when you add filters to create business analytics charts and graphs.

The following are the valid filter values for some of the system indicator columns:

1. Is Recoverable
 - **Y**: Represents Yes
 - **N**: Represents No
2. All other system indicator columns starting with "Is ..."
 - **0**: Represents false
 - **1**: Represents true
3. Due Status
 - DUE SOON
 - MISSED DUE
 - ON TRACK
 - OVERDUE
4. Process Deployment Status
 - **-1**: refers to un-deployed status
 - **0**: refers to retired status
 - **1**: refers to deployed status
5. Process Instance Status
 - ABORTED
 - ACTIVE
 - COMPLETED
 - FAULTED
 - SUSPENDED
6. Activity Instance Status
 - ABORTED
 - ACTIVE
 - COMPLETED
 - FAULTED
 - MOVED
 - SUSPENDED
7. Assignment state
 - ACQUIRED

- ASSIGNED
 - COMPLETED
 - ERRORRED
 - EXPIRED
 - DELETED
 - WITHDRAWN
 - SUSPENDED
- 8. Task State**
- ALERTED
 - ASSIGNED
 - COMPLETED
 - ERRORRED
 - EXPIRED
 - DELETED
 - INFO_REQUESTED
 - WITHDRAWN
 - SUSPENDED

Creating Analytics with Oracle Business Intelligence Cloud Service

Oracle Business Intelligence Cloud Service (Oracle BI Cloud Service) enables you to create, manage, and deploy analytics data to create your own dashboards. These dashboards are based on business data and allow you to get a broader and better perspective of your business processes. See About Oracle BI Cloud Service in *Using Oracle Business Intelligence Cloud Service*.

Integrating Oracle BI Cloud Service with Process enables you to archive business analytics from Process into Oracle BI Cloud Service. This integration lets you explore and access your archived data in Oracle BI Cloud Service. Once the archive is complete, you can sign in to Oracle BI Cloud Service and create charts or graphs by selecting data from multiple tables and creating business analytics reports as per your need. See Exploring Your Content and Adding Your Own Data in *Using Oracle Business Intelligence Cloud Service*.

The business analytics reports that you create in Oracle BI Cloud Service help you detect errors in your process and enable you to resolve them. You can also create your own dashboards. See Creating Analyses in *Visualizing Data and Building Reports in Oracle Analytics Cloud*.

How do I integrate with Oracle Business Intelligence Cloud Service?

Before you or any users can use and access Oracle Business Intelligence Cloud Service (BI Cloud Service), an administrator must configure settings in both Oracle Business Intelligence Cloud Service and Process.

Pre-requisites:

To create the integration between BI Cloud Service and Process, the administrator must ensure:

- The user is subscribed to both BI Cloud Service and Process.
- Analytics data is being captured in Process Database.
- User in BI Cloud Service has the required role for creating/updating tables in BI Cloud Service.

How does it work?

- [Configuring BI Cloud Service Settings in Process](#)
- [Scheduling Analytics Data Archive and Purge](#)
- [Creating Charts and Graphs in Oracle BI Cloud Service](#)

Configuring BI Cloud Service Settings in Process

As the administrator, you must also configure the connection between your Process and your Oracle BI Cloud Service. You need to enter information such as the URL and sign-in credentials for your Oracle BI Cloud Service. You can also add a unique table name prefix for each of the instances you archive.

To configure the settings in Process:

1. Go to the Home page, click **Configure**.
2. Click **Services**.
3. In the **Business Intelligence Cloud Service** section, enter the following information:
 - **URL**: the web address of your Oracle Business Intelligence Cloud Service, for example, <https://mybicloudservice-myidentitydomain.analytics.us1.oraclecloud.com>
 - **Identity Domain**: the name of the identity domain that your Oracle Business Intelligence Cloud Service belongs to.
 - **Database Table Name Prefix**: unique prefix for the table names in the BI Cloud Service Database.

 **Note:**

If you have only one BI Cloud Service instance and multiple Process instances, then for each instance, the table name in the BI Cloud Service database can be unique with a prefix. However, this is an optional attribute and if you don't enter a prefix, the default prefix "PCS" will be used in the table names.

- **User and Password:** the account credentials for a user who has access to your Business Intelligence Cloud Service. This user account is used for testing the connection between the services and for transferring the analytics data to BI Cloud Service.

4. Click **Test.**

Whenever you make any changes to the configuration settings, it's a good idea to verify that the values you entered are correct. You want to confirm that a successful connection has been established with Oracle Business Intelligence Cloud Service.

Review the test results, which may include messages, errors, and warnings.

5. Select one of the following options to continue:

- If there are any errors or warnings, make the necessary changes and then click **Test** again to verify the new values. Repeat the test each time you change the settings.
- If the connection test is successful, click **Save** to save the configuration settings.
- If you want to cancel and return to the last-saved values, click **Revert**.

Oracle BI Cloud Service is successfully configured in Process. To archive process analytics, see [Scheduling Analytics Data Archive and Purge](#) in *Using Oracle Business Intelligence Cloud Service*.

Creating Charts and Graphs in Oracle BI Cloud Service

After you enable your BI Cloud integration, the archived analytics data gets saved in different tables in Oracle BI Cloud Service:

- PCS_<Application Name>_ACTIVITY
- PCS_<Application Name>_PROCESS
- PCS_PROCESS
- PCS_ACTIVITY
- PCS_TASK
- PCS_ASSIGNMENT
- PCS_COMPOSITE_DEFINITION
- PCS_PROCESS_DEFINITION
- PCS_ACTIVITY_DEFINITION
- PCS_TASK_DEFINITION
- PCS_ROLE_DEFINITION

Note:

If the name of the application exceeds 30 characters, then the table name for the same gets shortened in Oracle BI Cloud Service. The data is stored in tables with names prefixed with "PCS_" by default.

An administrator can sign in to Oracle BI Cloud Service, and create charts and graphs using the data from these tables. You can select the data from multiple tables and create a business analytic report as per your need. See Creating Analyses and Building Dashboards in *Visualizing Data and Building Reports in Oracle Analytics Cloud*.

Note:

If you switch to a new Oracle BI Cloud Service instance, you can migrate your data from the old system to the new one. See Downloading, Uploading, and Migrating Snapshots in *Preparing Data in Oracle Business Intelligence Cloud Service*.

Configuring Application Settings

You can configure the settings for all your applications.

Go to the Home page, and click **Configure**.

Option	Description
Oracle Integration Cloud Service	Under Services, create a connection to Oracle Integration Cloud Service to start creating ICS integrations.
Oracle Content and Experience Cloud	Under Services, create a connection to Oracle Content and Experience Cloud so that end users can collaborate using documents and conversations.
Oracle Business Intelligence Cloud Service	Under Services, create a connection to Oracle Business Intelligence Cloud Service to archive analytics data to BI Cloud Service and create charts and graphs using the Process Analytics data.
Notification Service	Under Services, configure e-mail (human task) notifications.
Runtime Settings	Under Runtime Settings, configure the process audit level and schedule a daily recovery for process instances that encountered a remote fault while invoking a web service or expired timer messages and edit the logger's settings.
Archive	Archive process instances on demand or schedule instances archive, or schedule analytics archive to back up your data from one or more applications
UI Customization	Configure the UI to use your own logo, title, and background image. Update time zone settings and hide comments as needed.
Configure Credentials	Manage runtime keystore credentials for web and REST services. Upload, update, or delete credentials as needed.
Manage Certificates	Manage runtime security certificates for message protection. Upload, update, or delete certificates as needed.

Option	Description
Manage Roles	Assign roles to users and groups. If needed, create, modify, and remove roles.
Notification Log	View details and resend email notifications sent for human tasks.

Configuring Oracle Integration Cloud Service

Before you or any users can access the Oracle Integration Cloud Service, you must configure settings in Oracle Process Cloud Service. Only a user with administrator privileges can establish a connection between your Oracle Process Cloud Service and your Oracle Integration Cloud Service. As an administrator, you need to enter information such as the URL and sign-in credentials for your Oracle Integration Cloud Service.

To configure the settings for Oracle Integration Cloud Service:

1. Sign in to Oracle Process Cloud Service.
 2. On the Home page, click **Configure**.
 3. Click **Services**.
 4. In the **Integration Cloud Service** section, enter the following information:
 - **URL**: the web address of your Oracle Integration Cloud Service. Your service administrator receives a Welcome to Oracle Cloud email when the service is ready to use. The email has the URL for your Oracle Integration Cloud Service. For example, <https://myICSServer.myICSdomain.com/ics>.
 - **User and Password**: the account credentials for a user who has access to your Oracle Integration Cloud Service. This user account is used to test the connection between the services. It's also used during runtime to connect to the services and perform all the runtime operations. The user should have the IntegrationServiceDeveloper role in Oracle Integration Cloud Service.
 5. Click **Test**.
- Whenever you make any changes to the configuration settings, it's a good idea to verify that the values you entered are correct. You want to confirm that a successful connection has been established with Oracle Integration Cloud Service.
- Review the test results, which may include messages, errors, and warnings.
6. Select one of the following options to continue:
 - If there are any errors or warnings, make the necessary changes and then click **Test** again to verify the new values. Repeat the test each time you change the settings.
 - If the connection test is successful, click **Save** to save the configuration settings.
 - If you want to cancel and return to the last-saved values, click **Revert**.

Configuring Audit and Log Levels

You can select the type of messages you want to store in the audit trail and schedule a daily recovery for process instances that encounter a remote fault while invoking a

web service or expired timer messages. You can also recover the invoke or callback messages that were not delivered and resubmit them. You can use the logger setting to change the log levels of different loggers.

Configuring the Process Audit Level

To configure the process audit level:

1. Go to the Home page, and click **Configure**.
2. Click **Runtime Settings**.
3. In the **Process Runtime** section, select an audit level from the **Audit Level** drop-down list.

Available audit levels are:

- Production: logs all events but doesn't log the values used when assigning input or output values to data objects.
- Development: log all events and also logs the values used when assigning input or output values to data objects.
- Off: doesn't log any events.

Note:

If the audit level is **Production** or **Development**, the following activities will record the payload details:

- USER_TASK
- SERVICE_TASK
- RECEIVE_TASK
- SEND_TASK
- THROW_INTERMEDIATE_EVENT
- CATCH_INTERMEDIATE_EVENT
- BUSINESS_RULE_TASK
- START_EVENT END_EVENT

If the audit level is **Off**, then no audit information will be recorded.

4. Select a time to start and stop the scheduled recovery using the time editor.

The runtime environment uses the selected time and configuration to perform a daily scheduled recovery.

5. Enter the maximum number of instances to recover.
6. Click **Save**.

Configuring Logger Settings

As an administrator, you can use the **Logger Settings** section to change the log levels of different loggers and send an error report to Oracle if an error occurs. Logger levels include **Incident_Error**, **Error (Severe)**, **Warning**, **Notification (Info)**, **Notification (Config)**, and **Trace**.

Enabling Email Notifications

You can configure Process to use emails for human workflow notifications that are sent to the task assignee when events such as assignment and reassignment occur.

 **Note:**

Before enabling email notifications in runtime, make sure you customize the email notifications, for example, their content, template, attachments, and subject lines in design time, see [Customizing Notification Emails for Human Tasks](#). After enabling email notifications, you can view the notification logs and resend emails to all or some of the original recipients, see [Viewing and Resending Email Notifications](#).

To configure the human notification mode:

1. Go to the Home page, and click **Configure**.
2. Click **Services**.
3. Click **Infrastructure**.
4. Under **Notification Service**, select **Email** from the **Notification Mode** drop-down list, to use emails for human workflow notifications.
The email configuration details appear.
5. Optionally, you can configure the notification server by providing a new URL and the credentials to access it.
6. Enter the email address from which you want to send the notifications. In the **From** field, click **Register**, and then click **Save**.

Configuring Credentials for Web Services

You can use a credential to securely call a web service. You can add new credentials, make changes to existing credentials, or delete them.

You can manage credentials for a web service in both the environments. However, it's more common to create a credential when you're creating the connector to the web service. See [Creating a Web Service Connector](#).

Add a Runtime Credential

1. Go to the Home page, and click **Configure**.
2. Click **Manage Credentials**.
3. Click **Add new credential**.
4. Click the **Map** field and select **BPM WebServices Credential** for a web service credential.
5. Complete the following fields:
 - **Key**: an identifier for the web services credential
 - **User Name**: the name of the user to access the service.

- **Password:** the password to access the service. You must also enter this same password in the **Confirm Password** field.
6. Click **Save**.

You can also click **Revert** to revert your changes, or **Delete** to remove this credential.

After you create a credential, you can modify the user name, password, and description fields. To modify other settings, you must delete and recreate the credential.

Configuring Credentials for Basic Form REST Services

As the administrator, you may need to create custom headers to securely invoke a basic form REST Service. The custom header is a super set of Basic Auth support and uses a token for authentication instead of a user name and password. You can add or edit custom headers or delete them in the runtime environment.

For example, you have a drop-down list in your form, that contains a list of values you want the users to select from. This list of values needs to be retrieved from a secured REST service. So, you create a credential in the runtime environment. Meanwhile, in the design time environment, you create a form rule that retrieves the list of values from the secured REST service. If the REST call uses custom headers, you [manage header sets in Composer](#). You can use basic or OAuth authentication.

Adding Custom Headers and Authorization

1. Go to the Home page, and click **Configure**.
2. Under **Security**, select **Manage Credentials**.
3. Click **Add New Credential**.
4. In the Add Key dialog box, click the **Map** field and select **WebForm RESTServices Credential** for a basic form REST service credential.
5. Complete the following fields:
 - **REST URL:** Enter the URL for the REST service.
 - **Name:** Enter a token of the header.
6. Optionally, click **Add New Custom Header** to add a custom header. In the Add New Property dialog box, enter the following properties:
 - **Header Name:** Identifies the custom header property.
 - **Type:** Specify if the value should be secret (display as asterisks) or not secret (visible).
 - **Value, Confirm Value:** Enter the property value.
 - Click **Add**.
7. Click **Add Authorization** to add a basic or OAuth authorization.
8. To add a basic authorization, select **Basic Authorization** and complete the fields.
 - **User Name:** the name of the user to access the service.
 - **Password:** the password to access the service. You must also enter this same password in the **Confirm Password** field.

9. To add an OAuth authorization, deselect **Basic Authorization** and complete the fields.
 - **Auth Type:** Specify if the authentication type is secret or visible.
 - **Auth Key:** Enter the authentication key. You must also enter this same auth key in the **Confirm Auth Key** field.

Property Name	Value	Edit	Delete
customerName	your_org_name		
Authorization	*****		

10. Click **Add** when done adding custom headers and authorizations.

You can also click **Delete** to remove a selected header or authorization from the table.

Managing Security Certificates during Runtime

Certificates are used to validate an application's external web service connections when message security is applied. You can replace expired certificates and maintain a separate set of certificates for production use only.

See [Applying Message Security to Integrations](#) and [Managing Security Certificates during Design Time](#).

To view, add, update, or delete certificates:

1. Go to the Home page, and click **Configure**.
2. Click **Manage Certificates**.

Subject	Expires	Action
EMAILADDRESS=CN=...OU=PC8,O=Oracle,L=Redwood City,ST=San Francisco,C=US	Thursday, December 31, 2016 7:57:49 AM Pacific Daylight Time	
CN=Server Keystore,OU=Server,O=Server,L=BLR,ST=KA,C=IN	Saturday, September 24, 2016 11:31:59 PM Pacific Daylight Time	

Certificates for all applications display. If a certificate has expired, its icon displays a red caution (triangle) icon.

3. Filter the certificates list if needed.
4. Click a certificate's link to view its details.
5. To create a certificate, click **Upload Certificate** and enter an alias name. To upload from a file, select **Certificate File**, browse, and select the file. To paste certificate text, select **Certificate Content**, and paste in the **Certificate Content** field. Click **Upload**.
6. To delete a certificate, click its **Actions** icon and select **Delete**. To update a certificate, select **Update** and select a new certificate file or paste certificate contents.

Resolving Certificate Validation Exceptions

This section details the checks that you can perform if you encounter certificate validation exceptions when using Secure Sockets Layer (SSL) communication.

When you use SSL communication, you have to install security certificates, provided by a certificate authority, in Process Cloud Service or any other integration service. If any of the certificates are unverified or expired, or if the certificate chain is broken or unavailable, validation errors occur while accessing SSL endpoints. A common error encountered is:

```
javax.xml.soap.SOAPException: javax.xml.soap.SOAPException: Message send failed:  
sun.security.validator.ValidatorException: PKIX path building failed:  
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid  
certification path to requested target
```

You can perform the following simple checks to resolve the error:

- To check the validity of certificates and ensure proper SSL handshake, invoke Process Cloud Service from the command line using the following command:

```
curl -v https://<host>:443/ValexSBProject/<path>.
```

This command throws up an error if the SSL configuration is invalid or if the certificate chain is broken. If it displays errors, such as “*Peer's certificate issuer is not recognized*”, then the issue most likely lies on the service endpoint. If SSL handshake has errors, Process Cloud Service cannot consume a service through a service connector.

- Because SSL certificate chains are usually stored in the key store on a front-end server like Oracle Traffic Director (OTD), check if OTD is correctly configured and all certificates are available in the certificate chain.

The certificate chain usually consists of the certificate authority's root certificate, any intermediate certificate, and the issuer certificate. Multiple certificate chains in OTD will also result in validation errors.

Customizing the User Interface

You can configure the user interface to use a custom logo, title, and background image. You can also customize it in other ways, including showing dates based on the user's time zone and hiding the comments section.

To customize the User Interface:

1. Go to the Home page, and click **Configure**.
2. Click **UI Customization**.

3. Customize one or more of the available options:

- In the **Branding Logo** field, enter a URL to the image you want to use as the logo. For non-retina displays, use a maximum image size of 137px by 17px. For retina displays, use a maximum image size of 272px by 34px. This image appears in the upper-left corner of the page. By default, the page displays the Oracle logo. Supported image formats are PNG and JPG.
- In the **Branding Title** field, enter the text you want to use as the title. The title has a maximum of 40 characters, including spaces. This text appears in the upper-left of the page to the right of the logo.
- In the **Background Image Style** field, enter the CSS style that defines the background image for the application. The URL must not contain spaces. For example:
`background-image:url('http://www.example.com/my_background_image.jpg')`
- Select the **Show Dates in User's Time Zone** check box to display dates to users based on their system's time zone. By default, this setting is unchecked and all times are displayed based on server time zone.
- Select the **Hide comments if Oracle Social Network integration is enabled** check box to hide the comments section on the Task page and the Process tracking page. This option is valid only if you have integrated Oracle Content and Experience Cloud with Process, and you're using the conversations feature. See [How do I integrate with Oracle Content and Experience Cloud?](#)

4. Click **Save**.

To restore the default settings, click **Reset**, and then click **Reset to Default**. To revert your changes to the last saved version, click **Reset**, and then click **Reset to Last Saved**.

Configuring Oracle Content and Experience Cloud

Before users can access the documents and conversation features, an administrator must configure settings in both Oracle Content and Experience Cloud and Process.

- To read about the benefits of integrating with Oracle Content and Experience Cloud, see [Why should I integrate documents and conversations?](#)
- To learn more about how to configure and set up your services, see [How do I integrate with Oracle Content and Experience Cloud?](#)

Archiving and Purging Data

You can submit requests for an on-demand archive, a scheduled instances archive and purge, or a scheduled analytics archive to back up your process instances from one or more applications.

Topics:

- [About Archive and Purge](#)
- [Archiving On Demand](#)
- [Scheduling Instances Archive and Purge](#)
- [Scheduling Instance Purge Only](#)

- [Purging Transient Data](#)
- [Scheduling Analytics Data Archive and Purge](#)
- [Scheduling Analytics Data Purge Only](#)
- [Viewing Archive Requests](#)

About Archive and Purge

You can submit requests for an on-demand archive, a scheduled instances archive and purge, or a scheduled analytics archive to back up your process instances from one or more applications.

The data you archive gets saved in Oracle Storage Service. After the on-demand archive request is processed, you receive an email with a link to download the archived ZIP file. In the case of a single request, you receive the link to a single file. In the case of scheduled requests, there are multiple files and they're located in Oracle Storage Service.

If you enable the synchronization of analytics data to Oracle Business Intelligence (BI) Cloud Service, the data from the analytics archive also gets saved in Oracle BI Cloud Service. As an administrator, you can sign in to Oracle BI Cloud Service and create a detailed business analytics report using the data from different analytics tables.

You can also schedule an automatic purge from the Schedule Instances Archive page and be able to remove the BPM runtime information. The purge runs as a separate job on the database and ensures optimal performance. You can enable the archive and purge to start at the same time. Even though the archive and purge functions share the same calendar expression, they execute in different environments and operate on different data sets, and so they can safely operate independent of each other. Both archive and purge can be configured to run within a specific time limit. If the archive or purge exceeds the specified time limit, then the operation stops and picks up where it left off on the next scheduled run.

Purging Data Without Archiving

To purge instance data without archiving the data, select the **Purge only** option on the Schedule Instances Archive and Purge tab or the Schedule Analytics Archive tab in runtime administration.

File System Usage Maximum Limit

During instance archive and purge, instances collect in a staging area in the file system. A maximum limit to file system usage has been placed on the archive function when it's collecting the data to archive during instance archive and purge. When the staging area reaches the maximum limit, an archive ZIP file gets created and uploaded to the storage server. After upload, the staging location space is then cleared.

When Recoverable Faulted Instances Are Purged

If you have recoverable faulted instances that are not acted on and resolved, then after a minimum of 30 days, those instances are purged without being archived. However, if the **Purge Retention (Days)** field is set to more than 30 days, then the recoverable faulted instance retention will be set to match this greater retention time.

Purging Process Applications Not Yet Deployed

You can purge undeployed application instances, but you can't archive them. As a result, the On Demand Instances Archive tab, which has no purge option, only lists currently deployed application instances. If you want to purge application instances that haven't been deployed yet, then use the purge options on the Scheduled Instances Archive and Purge tab.

Cases Where Purging May Not Occur When Retention Day Reached

Purge won't run even if the Retention Day is reached in cases where all completed instances aren't archived for that day.

The archive has some built-in resiliency features in order not to compromise the file system when performing the export. In cases where there are many, and possibly large, process instances to be exported, the archive stops exporting when a pre-determined threshold is reached. At that point, the information exported so far is packaged to a ZIP file and uploaded to Oracle Cloud Storage Service. The archive picks up from where it left off the next time around by recording the date of the last item in the Archive Purge History table. Note that it's the Archive Purge History table that drives the purge. The last archive date is used to calculate the purge date (less the retention time). This behavior can give the impression that items aren't being purged on time or as expected because the date the purge operates on is not the run date of the purge process, but rather the last archive date.

Archiving On Demand

You can archive your process instances to a ZIP file. When the data is archived, the system sends an email with the link to the ZIP file. You can choose to send the email to your primary address or to an alternate address.

To submit a request for data archive:

1. On the Home page, click **Configure**.
2. In the side pane, click **Archive and Purge**.
3. Click the **On Demand Instances Archive** tab.
4. Select the state (**Active** or **Inactive**) and the status of the instances you want to archive.
 - If you select **Active**, then the available statuses are **All Active**, **Recoverable**, **Suspended**, and **Running**.
 - If you select **Inactive**, then the available statuses are **All Inactive**, **Completed**, **Failed**, and **Terminated**.
5. In the **Application** table, select the application instances you want to archive. To select more than one instance, press Shift while clicking the applications.
6. Specify the process information that you want to archive by selecting the appropriate check boxes:
 - **Include Audit Payload**: includes the instance payload (the values of the input and output parameters) for each activity.
 - **Include Audit Image**: includes the graphical view of the audit trail. The graphic shows where in the process flow the process instance is located at the time of the export.

- **Include Task History:** includes all the details of the lifetime of a task, such as assignments and reassignments.
 - **Include Rules History:** includes the history of running the business rules in the selected processes.
7. Optionally, to prevent unauthorized access to the archive file, select the **Encrypt Archive** check box. Enter and confirm the password you'll use to access the archive file.
8. By default, Process archives instance data and then sends the link to the exported ZIP file to the e-mail address specified in your My Services profile. To send the link to a different e-mail instead, select the **Use Alternate Email** check box, and then enter the e-mail address in the **Alternate Email** field.
- If your My Services user profile doesn't specify an e-mail address, then you must specify an alternative e-mail.
9. If you're archiving **Inactive** instances (that is, the **Instance State** field is set to **Inactive**), specify how to handle the instances after they're archived.
- To purge the **inactive** instances after they're archive, select the **Delete Instances After Export** check box.
 - If you're archiving any **inactive** instances with a status of **Completed**, then you can choose to keep the instances created after a certain date by selecting a date using the **Keep Instances Created After** time editor.
10. Click **Archive**.

Review the confirmation message for details about your archive request.

To check the status of your archive, expand the **Archive Requests** section on the page. When the archive process is finished, you can download the archive ZIP file using the **Completed** link in the **State** column. See [Viewing Archive Requests](#).

Remember that the link to the archive ZIP file is also sent to the email address that you specified in your My Services profile or in the **Alternate Email** field.

Scheduling Instances Archive and Purge

You can archive and purge your data automatically by creating a schedule that will perform actions according to either the time you select or the CRON expression you specify.

To schedule instances archive and purge:

1. On the Home page, click **Configure**.
2. In the side pane, click **Archive and Purge**.
3. Click the **Schedule Instances Archive and Purge** tab.
4. Select the **Archive and Purge** option.

If you don't need to archive instance data before purge, select **Purge only** and see [Scheduling Instance Purge Only](#).

5. Select the **Enable Archive and Purge** check box. The configuration fields become available.

Note that:

- If you want to switch to the **Purge only** option, then you must first deselect the **Enable Archive and Purge** check box.

- Deselecting the **Enable Archive and Purge** check box and then saving your change terminates all previously configured schedules.
6. Configure the schedule for this archive. You can use either the time fields or the advanced scheduling interval.
- Use the time fields to specify when (how often, what day, and what time) the archive occurs. For example, to schedule an archive for *every week on Friday at 03:20*, for **Every**, select **Week**, for **on** select **Friday**, and for **at**, select **03** for hour and **20** for minutes.
 - Use the advanced scheduling option to specify a CRON expression. Select the **Use Advanced Scheduling Interval** check box and enter a valid CRON expression. Click the adjacent help icon for CRON details and examples.

 **Note:**

Enabling the **Use Advanced Scheduling Interval** check box automatically disables the settings in the time fields.

7. Expand the **Configure Archive Content** section. Specify the process information that you want to archive by selecting the appropriate check boxes:
- **Include Audit Payload**: includes the instance payload (the values of the input and output parameters) for each activity.
 - **Include Audit Image**: includes the graphical view of the audit trail. The graphic shows where in the process flow the process instance is located at the time of the export.
 - **Include Task History**: includes all the details of the lifetime of a task, such as assignments and reassignments.
 - **Include Rules History**: includes the history of running the business rules in the selected processes.
8. In the **Archive Job Timeout** field, enter the number of minutes that the archive is allowed to run.
9. In the **Failure Notification Address** field, enter the email address that all failure notifications will be sent to.
10. If you want to delete the instance data after the archive has finished, select the **Purge Archived Data** check box.
- In the **Purge Retention** field, enter the number of days that data should be retained for after archiving. Once the number of days is complete, the data gets purged.
 - In the **Purge Job Timeout** field, enter the number of minutes the purge will be allowed to run.
11. Click **Save**.

Review the confirmation message for details about your request.

You can expand the **Archive Requests** section to view the status of your scheduled archives. See [Viewing Archive Requests](#).

Scheduling Instance Purge Only

You can purge your data automatically by creating a schedule that deletes the data according to either the time you select or the CRON expression you specify.

To schedule instance purge:

1. On the Home page, click **Configure**.
2. In the side pane, click **Archive and Purge**.
3. Click the **Schedule Instances Archive and Purge** tab.
4. Select the **Purge only** option.

If you need to archive the data before it's purged, then select **Archive and Purge**, and see [Scheduling Instances Archive and Purge](#).

5. Select the **Enable Purge** check box. The configuration fields become available.

Note that:

- If you want to switch to the **Archive and Purge** option, then you must first deselect the **Enable Purge** check box.
 - Deselecting the **Enable Purge** check box and then saving your change terminates all previously configured schedules.
6. Configure the schedule for this purge request. You can use either the time fields or the advanced scheduling interval.
 - Use the time fields to specify when (how often, what day, and what time) the purge occurs. For example, to schedule a purge for *every week on Friday at 03:20*, for **Every**, select **Week**, for **on** select **Friday**, and for **at**, select **03** for hour and **20** for minutes.
 - Use the advanced scheduling option to specify a CRON expression. Select the **Use Advanced Scheduling Interval** check box and enter a valid CRON expression. Click the adjacent help icon for CRON details and examples.

Note:

Enabling the **Use Advanced Scheduling Interval** check box automatically disables the settings in the time fields.

7. In the **Purge Retention** field, enter the number of days for which data should be retained.
8. In the **Purge Job Timeout** field, enter the number of minutes the purge will be allowed to run.
9. Click **Save**.

Purging Transient Data

As the processes of an application progress, their information gets saved in the database. However, the usefulness of some of this data is short-lived. Such form of data that is not required and not referred to anymore is called **transient data**. You can

remove the obsolete transient data for completed and active process applications, and reclaim the space.

To purge transient data:

1. On the Home page, click **Configure**.
2. In the side pane, click **Runtime Settings**.
3. Expand the **Purge Transient Data** section.
4. Click **Purge Transient Data** to clean up the transient runtime data for completed processes.

When you click **Purge Transient Data**, the purge job is submitted to the background. It may take some time for the job to complete. While the job is in progress, the **Purge Transient Data** button disappears. Click the **Refresh** icon  to check the status of the job.

The **Purge Transient Data** button reappears only when the purge job is complete and only if there is further transient data to be removed.

Scheduling Analytics Data Archive and Purge

Archive and purge your analytics data automatically by creating a schedule that performs actions according to the time you select or the Cron Entry you use. You can also enable synchronization of data to Oracle Business Intelligence Cloud Service (BI Cloud Service) after configuring runtime settings.

You can use REST APIs to work with analytics archive data. See Oracle Process Cloud Service REST API version 4.0.

To schedule analytics archive and purge:

1. On the Home page, click **Configure**, then **Archive and Purge** in the side pane. Click the **Schedule Analytics Archive** tab.
2. In the top field, ensure that **Archive and Purge** is selected. If you don't need to archive analytics data before purge, select **Purge only** and see [Scheduling Analytics Data Purge Only](#).
3. Select the **Enable** check box for the action you selected in the previous step. The configuration fields below become available. (This check box must be deselected to change actions.)

Note:

Deselecting the **Enable** check box and saving terminates all previously configured schedules.

4. In the Schedule fields, set a schedule using the time fields or the advanced scheduling interval.
 - Using the time fields, set an archive time. For example, to schedule an archive for every **Week on Friday at 03:20 hours**, for **Every**, select **Week**, for **on** select **Friday**, and for **at**, select **03** for hour and **20** for minutes.
 - Use the advanced scheduling options to specify a **Cron Entry**. Select the **Use Advanced Scheduling Interval** check box and enter a valid **Cron Entry**. Click the adjacent help icon for CRON details and examples.

 **Note:**

Enabling **Use Advanced Schedule Interval** disables the time filters.

5. To receive failure notifications, enter your email address in the **Failure Notification Address** field.
6. To integrate with BI Cloud Service, select the **Synchronize Archive to Business Intelligence Cloud Service** check box. Disabling this option terminates any current scheduled analytics archive requests.

Before synchronizing archived analytics data from Process to BI Cloud Service, you must configure settings in runtime. See [Configuring BI Cloud Service Settings in Process](#)
7. To archive and also purge instances, select the **Purge Archived Data** check box and complete the purge fields below that become available.
 - In the **Purge Retention** field, enter the number of days for which data should be retained for after archiving. Once the number of days is complete, the data gets purged.
 - In the **Purge Job Timeout** field, enter the number of minutes the purge will be allowed to run.
8. Click **Save**.

 **Note:**

After you schedule an analytics archive, some records may not be archived to the Cloud Storage Service or get transferred to BI Cloud Service. In such cases, the missing records are picked up during the next archive cycle.

A confirmation message appears with the request ID.

You can view archive requests in the **Archive Request** table. See [Viewing Archive Requests](#).

Scheduling Analytics Data Purge Only

You can purge your data automatically by creating a schedule that purges the data according to the time you select or the Cron Entry you use.

To schedule analytics data purge:

1. On the Home page, click **Configure**, then **Archive and Purge** in the side pane. Click the **Schedule Analytics Archive** tab.
2. In the top field, select **Purge only** if you don't need to archive analytics data before purge. If you need to archive, select **Archive and Purge**, and see [Scheduling Analytics Data Archive and Purge](#).
3. Click the **Enable Purge** check box.

Note:

Deselecting the **Enable** setting and saving terminates all previously configured schedules.

4. In the Schedule fields, set a schedule using the time fields or the advanced scheduling interval.
 - Using the time fields, set a purge time. For example, to schedule a purge for every **Week on Friday at 03:20 hours**, for **Every**, select **Week**, for **on** select **Friday**, and for **at**, select **03** for hour and **20** for minutes.
 - Use the advanced scheduling options to specify a **Cron Entry**. Select the **Use Advanced Scheduling Interval** check box and enter a valid **Cron Entry**. Click the adjacent help icon for CRON details and examples.

Note:

Enabling **Use Advanced Schedule Interval** disables the time filters.

5. In the **Purge Retention** field, enter the number of days for which data should be retained.
6. In the **Purge Job Timeout** field, enter the number of minutes the purge will be allowed to run.
7. Click **Save**. (Click **Reset** to clear the fields.)

Viewing Archive Requests

When you submit a request for an on-demand or scheduled archive or purge, the Archive Requests table displays the current status of the job.

To view the Archive Requests table, go to the Home page, click **Configure**, click **Archive and Purge** in the side pane, and then expand the **Archive Requests** section.

State	Description
Pending	The archive request has been submitted for processing, but it hasn't been picked up by the scheduler.
Running	The archive is in progress.
Completed	<p>The archive is complete.</p> <p>For the single archive, the system sends a notification email that the archive has finished. The email includes a link to download the archived file. If there are any errors, then the email includes details about the problem. Contact your administrator for assistance in resolving any issues.</p> <p>For the scheduled instance archive, the data is stored in Oracle Storage Service.</p>
Downloaded	The user has downloaded the archive file at least once.
Terminated	The request was canceled or stopped, most likely due to a server restart. May also occur when the schedule gets changed.
Failed	An exception occurred while executing the job. To get details, view the server log file or use the REST APIs.

 **Note:**

A scheduled request maintains its job ID and one row, and the state alternates between **Pending** and **Running**. The scheduled request gets **Terminated** only when the schedule gets changed. In that case, the scheduled request gets a new job ID. The request will again go into the **Pending** and **Running** states.

Troubleshooting the Runtime Administration

The following troubleshooting tips apply to the runtime environment.

The View Dashboards and Configure icons aren't showing up in my Home page

Verify that you're assigned the administrator role for Process. It's required to access these components. See [About Process Roles and Users](#).

How do I find out who is currently assigned to a specific process instance?

In the Instances page, locate the instance using the filters or by searching. Select it to view its details. In the Details pane, expand the **More Information** section. The Assigned To field indicates the current assignee for that process instance. See [Tracking Process Instances](#).

How do I recover a failed process instance?

Locate the instance using the filters in the Instances page. If you're an administrator, you can also use click the alerts on the home page. Select the instance to view its details. Click the **Alter Flow** button to edit the value of the process instance data objects and resume it. See [Altering the Flow of a Process Instance](#).

How do I reassign a task that a user assigned to the wrong person?

If you or a user assigned a task to the wrong person by mistake, you can locate the process instance and reassign the task. Use the filters in the Tracking page to locate the instance, select it, and reassign it to a different user. See [Can I reassign my tasks to someone else?](#).

Managing the Design-Time Environment

As an administrator, you can administer and manage the design-time environment.

Topics

- [Administering Spaces, Applications, and the Player](#)
- [Managing Environments and Activating Applications](#)

Administering Spaces, Applications, and the Player

In the design-time environment, manage spaces, applications, and the player.

Topics

- [Administering Spaces](#)
- [Unlocking and Deleting Applications](#)
- [Deleting QuickStart Apps from the Gallery](#)
- [Enabling the Application Player](#)

Administering Spaces

Check the status of a space (private or shared), control permissions to a space, or delete a space. As an administrator, you may need to perform these actions if, for example, the owner of the space is on vacation.

Go to the Home page, click **Develop Processes**, and then click **Administration** to display the Administration pages. The Spaces Administration page displays by default.

The screenshot shows the 'Administration' section of the Oracle Application Server interface. At the top, there are four main icons: 'Spaces' (blue circle with a white gear), 'Applications' (orange circle with a white cloud), 'QuickStart Apps' (purple circle with a white icon), and 'Player' (green circle with a white icon). Below this, the 'Spaces Administration' page is displayed. It features a header with 'View ▾', 'Search', and 'Detach' buttons. The main area is a table listing four spaces:

#	Space	Creator	Status	Actions
1	Finance	JAUSTEN	Online	Details Lock Delete
2	HR	ISTONE	Online	Details Lock Delete
3	My Space	JAUSTEN	Online	Details Lock Delete
4	Payroll	CDOYLE	Online	Details Lock Delete

To control permissions to a space, click **Share**  in the **Actions** column. In the Share -space-name dialog box, perform any of the following actions:

- To add a user to the space, enter or search for the user, select a role, and click **Share**.
- To change the user's role in the space, select the user from the table, click **Edit Role**, select a new role, and click **OK**.
- To delete a user from the space, select the user from the table, click **Delete**, and click **Yes**.

To delete a space, click **Delete space and content**  in the **Actions** column. Note that deleting a space deletes the space and all the applications it contains.

! Important:

Contents can't be recovered once the space is deleted.

Unlocking and Deleting Applications

Use the **Applications** option on the Administration page to unlock a shared application locked by another user, or to delete a shared application.

Shared applications are automatically locked when a user opens the application in **Edit** mode to prevent other users from editing an application at the same time. As an administrator, you may have to release a lock if, for example, a user has forgotten to close an application before going on vacation.

To unlock a shared application, click **Unlock Application**  in the **Actions** column.

⚠ Caution:

If you unlock an application that has unpublished changes, the changes are lost and can't be recovered.

As an administrator, you can delete shared applications. A private application, on the other hand, can be deleted only by one of its owners.

To delete a shared application, click **Delete Application**  in the **Actions** column. The application is removed from the Process repository.

You can't delete an application that is locked.

Deleting QuickStart Apps from the Gallery

Users can rapidly build a new application based on any QuickStart App listed in the gallery. They can personalize the application to fit their business needs and then activate it. As an administrator, you need to monitor and manage the list of QuickStart Apps available in the gallery. For example, you may want to remove a QuickStart App

that is no longer used, remove one until an updated version is available, or simply remove one because it's a duplicate.

Deleting a QuickStart App removes the QuickStart App from the gallery only. Users will no longer be able to select the QuickStart App to create an application. However, the source application is still available on the design-time Home page. You can promote it back into the gallery at any time.

To delete a QuickStart App from the gallery:

1. Go to the Home page, click **Develop Processes**, and then click **Administration** to display the Administration pages.
2. Click **QuickStart Apps**.

This page lists all your QuickStart Apps that have been promoted to the gallery. It doesn't include the standard QuickStart Apps, such as Loan Application or Travel Approval, provided by Oracle. You can't delete the standard QuickStart Apps from the gallery.

3. Click **Delete**  in the Actions column to remove the QuickStart App from the gallery.

Enabling the Application Player

You can use the application player to test the behavior of a business process at design time. You can test a process using different user IDs without having to explicitly activate the application and test the business process using the runtime environment.

When testing a business process, the application player activates a version of your application to a special partition in Process. As a result, the player runs your business process using the same environment as your normally activated applications.

To enable the application player, you must provide the credentials for your Process administrator. These credentials are the user name and password of any user with administrator privileges.

After you provide the administrator credentials, any user who has editing privileges for a process application can run the application player.

When starting the player, the design-time environment **test activates** the application to the cloud using the specified credentials. When the current user wants to perform an activity or a task as another user, the design-time environment also uses these credentials.

To enable the application player:

1. Go to the Home page, click **Develop Processes**, and then click **Administration** to display the Administration pages.
2. Click **Player**.
3. Enter the credentials for an administrator of your Process.
4. Click **Save**.

After enabling the application player, users that have editing privileges can use the player to test the behavior of their business processes.

 **Note:**

Clicking **Clear** removes the credentials and as a consequence, the application player is disabled and no longer available to users. The administrator can also use this procedure to simply change the credentials (change the password).

Want to learn more about using the application player? See [Playing Processes and Testing Applications](#).

Managing Environments and Activating Applications

From the Manage Deployed Applications page in the design-time environment, administrators can activate applications, configure environments and activation permissions, and manage activations.

Go to the Home page, click **Develop Processes**, and then click **Management** in the global toolbar, and enter your credentials. The **Management** link is visible only if you have administrator privileges.



The Manage Deployed Applications page displays information about your servers, such as current status and activate permissions (private or open), and information about each activated application, such as name, revision ID, and current status. If an application was activated from a particular snapshot, then the Application Name column includes both the application name and the snapshot name.

Manage Deployed Applications						
		Application Name	Revision id	Mode	Status	Deployed Date
My Server	Private	LoanApplication	1.0	Active	On	11/13/2015 8:05 AM
Remote Server	Not configured	NotifyApp	2.0	Active	On	11/12/2015 1:02 PM
	Click for more information	LendingSubmission	1.0	Active	On	11/12/2015 1:08 PM
		BelgiumApprovalProcess	88	Active	On	11/12/2015 6:35 PM
		ExclusiveGtwDefaultPath	1.0	Active	On	11/12/2015 1:26 PM
		FaultApplication	1.0	Active	On	11/12/2015 2:05 PM

From the Manage Deployed Applications page, you can:

- Configure activation permissions
- Activate applications
- Perform specific actions, such as deactivate, retire, or shutdown an activated application

- Configure a remote server
- Enable message security and manage certificates

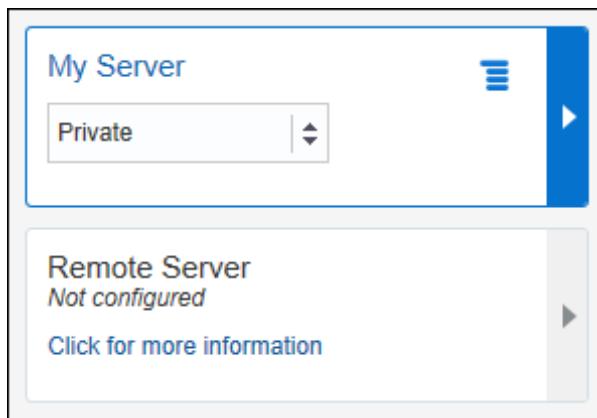
Configuring the Activation Permissions

Let's talk about the activation strategy. Do you want just your administrators to be able to activate an application? Developers can create applications and test activate them to a sandbox environment, but they have to ask administrators to activate the final application to production. On the other hand, do you want developers to be responsible for activating their applications?

Think about how you want to handle this and what approach is best for your organization. Also, if you have more than one instance of an Oracle Process Cloud Service, then different environments can have different strategies. For example, one instance might be for building, modeling, and testing applications. This instance is **open** so developers and administrators can activate applications. The second instance is only for production applications. This instance is **private**, and only administrators can activate applications.

To configure the activation permissions:

1. Go to the Home page, click **Develop Processes**, and then click **Management** in the global toolbar.



2. For each server, use the drop-down list to select the permission for activating applications to the server.
 - **Private**: Only administrators can activate applications to the server.
 - **Open**: Administrators and developers can activate applications to the server.

Activating Applications

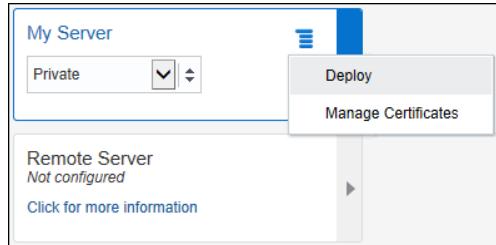
If you have administrator privileges, then you can activate applications to a production or testing environment from the Manage Deployed Applications page in design time.

Note:

To quickly back up your application, export it as an EXP file, and save it locally. You can then import it at anytime from the Process Applications page. See [Importing and Exporting Applications and Snapshots](#).

To activate an application:

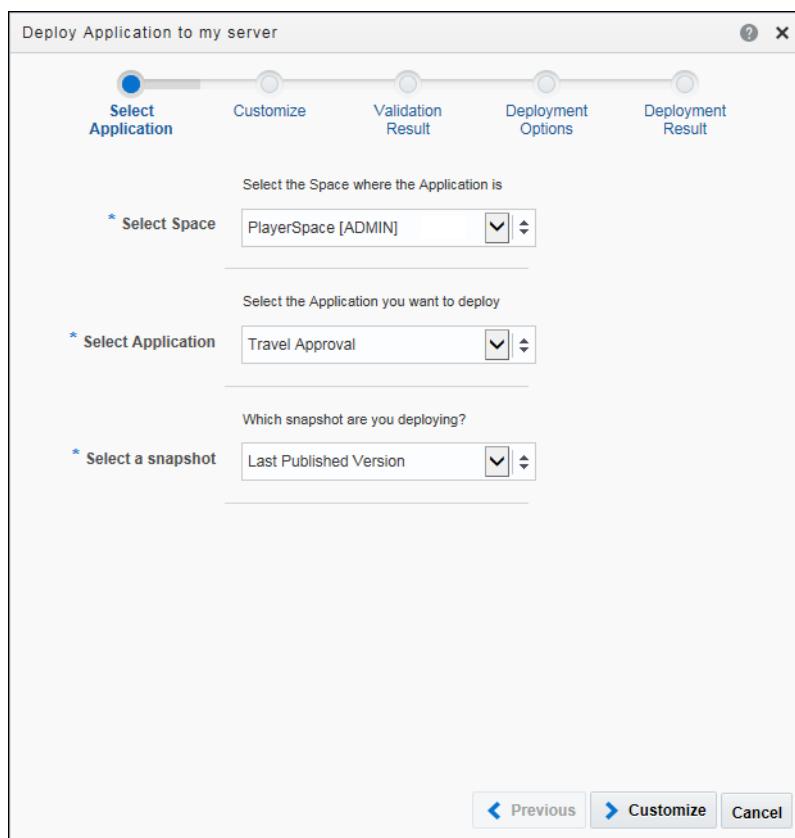
1. Go to the Home page, click **Develop Processes**, and then click **Management** in the global toolbar.
2. Click **Options**  for the server you want to activate to, and select **Deploy**.



The deploy wizard opens. It will walk you through the activation process from selecting your application to setting options and viewing results. Continue reading if you want a preview of the activation process.

Selecting the Application to Activate

In the deploy wizard, you select the space where the application is located, and the name and version of the application you want to activate. The available versions of the application are the **Last Published Version** and all snapshots that have been created.



The screenshot shows the 'Deploy Application to my server' wizard. The current step is 'Select Application'. The interface includes a progress bar with five steps: 'Select Application' (highlighted in blue), 'Customize', 'Validation Result', 'Deployment Options', and 'Deployment Result'. Below the progress bar, there are three main configuration sections:

- Select the Space where the Application is**: A dropdown menu labeled 'Select Space' with the value 'PlayerSpace [ADMIN]'.
- Select the Application you want to deploy**: A dropdown menu labeled 'Select Application' with the value 'Travel Approval'.
- Which snapshot are you deploying?**: A dropdown menu labeled 'Select a snapshot' with the value 'Last Published Version'.

At the bottom of the wizard, there are navigation buttons: '< Previous', 'Customize', and 'Cancel'.

 **Note:**

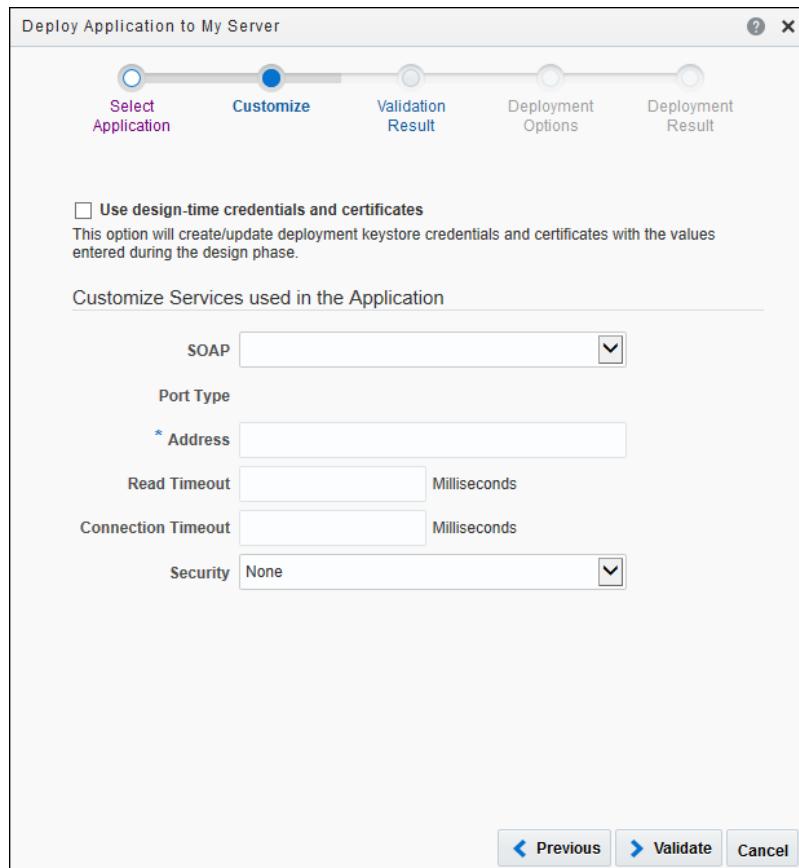
As an administrator, you can select from all applications inside any space.

Customizing Web Services and Properties

Optionally, you can customize the services used in the application as well as other properties before activation. Your changes will modify the original application and will be activated to the server. Your changes aren't published to the repository so you won't see them when editing the application.

 **Note:**

You can skip the Customize page. You don't have to select any of these optional settings.



When designing the application, you can define authentication keys to use when activating the application to the test environment. There are two different sets of credentials:

- One set for design time
- One set for runtime

This helps to keep the two environments completely isolated from one another. Nobody can change or use a runtime credential when designing or testing the application.

When the Customize page opens, it displays a list of connectors that are used in the processes. After you select a connector, you can view and edit its advanced options and security policy. Depending on the security policy, you must either select or create a security credential, and select or add a certificate alias.

On the Customize page, you're defining the runtime credentials. Remember that. You can select the **Use design-time credentials and certificates** check box to determine which set of certificates and credentials display in the Certificate Alias and Keystore Credential fields. Selecting the check box is a shortcut. It's a quick way to add the credentials and certificates that were defined at design time, and use them to create the runtime credentials.

If you select one of the design-time credentials to use (for example, `myKey`), then that credential is copied and a new runtime credential is created in the credential store. There are now two versions of this credential key: one for design time; the other for runtime. If you change the *username* and *password* during the activation wizard, then your change only affects runtime credentials. If you change the *username* and *password* in the design-time web service connector wizard, then your change only affects design-time credentials.

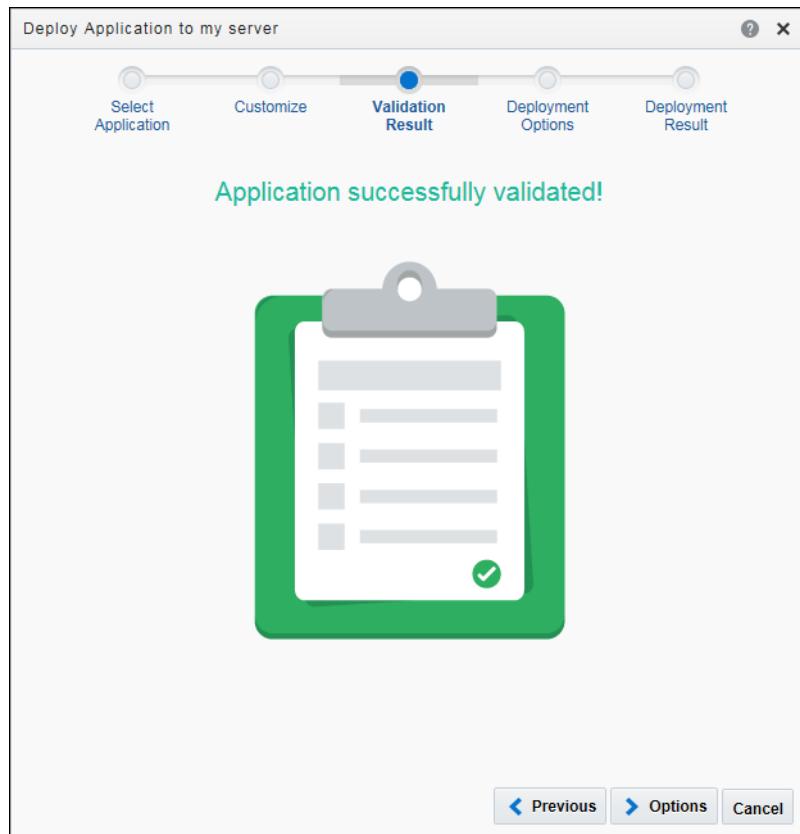
The credential and certificate are created or updated when you click **Deploy**.

Want to learn more about web services, credentials, and certificates? See [Creating a Web Service Connector](#).

Reviewing Validation Results

The Validation page displays the result of the validation triggered by the **Validate** button on the Customize page.

- **Validation Success:** A message indicating validation is successful displays. However, although the validation may be successful, a table with a list of warning messages may display.



If no warnings exist, a placeholder image is displayed. The **Options** button is enabled, as shown.

- **Validation Failed:** A message indicating validation failed displays as well as a table with the validation errors. The **Options** button is inactive.

When validation fails, you can't continue until all errors have been fixed and validation is successful.

Specifying Deployment Options

On the Deployment Options page, you must enter:

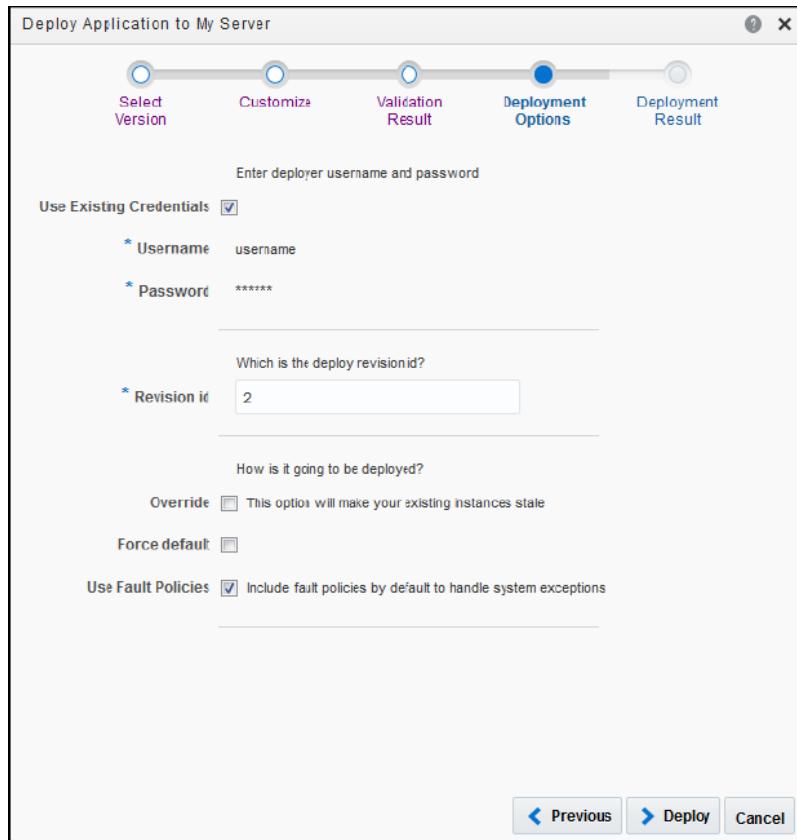
- The name and password of a user who has activation privileges. By default, the credentials for the user currently signed in are already entered.
- A revision ID, such as 1 or 1.0, to assign to this activation.

Optionally, you can select whether this version of the application should override the previous version and whether it should be the default version.

Note:

If you select to override the previous version, existing instances become stale. If you select the force default check box, only the default (primary) version only of each application is used.

In certain cases, it may be useful to disable automatic importing of system or runtime exceptions so that they can be handled at the process level. By default however, the **Use Fault Policies** field is selected and system faults are NOT caught by the process.



Viewing Deployment Results

When you click **Deploy**, the application is activated to the selected environment.

- *Deployment Success:* A message indicating the application activated successfully displays. The active application is added to the Deployed Applications table for the environment.
- *Deployment Failed:* A message indicating an error displays with more specific information about the error.

Managing Active Applications

After an application is activated, you might need to change its status or mode. For example, you might need to shut down an application temporarily or retire an application indefinitely.

Go to the Home page, click **Develop Processes**, and then click **Management**.

Click **Options**  in the Actions column.

You can select to retire (or activate), deactivate, and shut down (or start up) an application. You can also view the web services exposed and consumed by the application.



Deactivating an Application

Deactivating completely removes the application from the cloud. The application is no longer available to users and no longer listed on the Manage Deployed Applications page.

Retiring and Activating Applications

Retire and activate are opposite actions. Retiring an application retires an active application and you can no longer create a new instance from that application version. Instances that are already created continue to run and new requests for existing instances are accepted. Activating an application brings it back from the retirement stage. The Manage Deployed Applications page reflects the new status in the **Mode** column (**Active** or **Retired**).

Activate activates a retired application revision. Note the following behavior with this option:

- All applications are automatically active when activated.
- Other revisions of a newly activated application remain active (that is, other revisions aren't automatically retired). If you want, you must explicitly retire them.

Shutting Down and Restarting Applications

Shutting down an application is possible only after starting an application. Once an application is shut down, any request related to the application is rejected and no new request can be processed. All instances are stopped. You can restart the active application revision that was shut down. This action enables new requests to be processed (and not be rejected). No recovery of messages occurs. The Manage Deployed Applications page reflects the new status in the **Status** column (**On** or **Off**).

Note:

Shutdown shuts down a running application revision. Any request (initiating or a callback) to the application is rejected if the application is shut down. However, the behavior differs based on which component is used. For example, for a web service request, it's rejected back to the caller.

Viewing Information About Web Services

To view information about the web services used by a active application, click **Web Services**.

- The Exposed tab displays the web services URLs that the active application is exposing.
- The Consumed tab displays the web services that the currently active application connects to.

Configuring a Remote Server

In the design-time environment, the local environment (My Server) is automatically set up and configured for you. If your company purchases a second instance of Oracle Process Cloud Service, then you must set up and configure that instance manually.

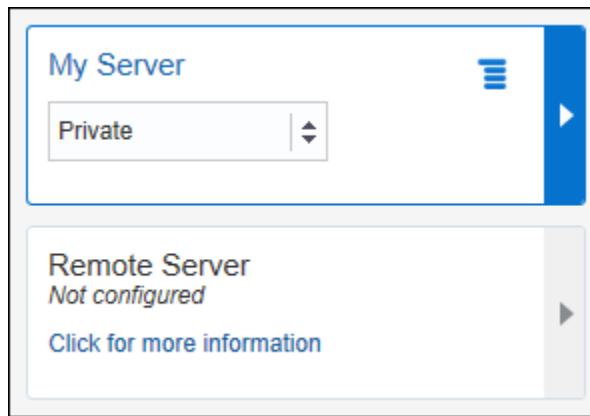
You can set up and configure your server environments from the Manage Deployed Applications page. If the remote server isn't currently configured, then you can manage and activate applications to the local environment (My Server). The remote server is disabled.

Before you try to configure a remote server, note that:

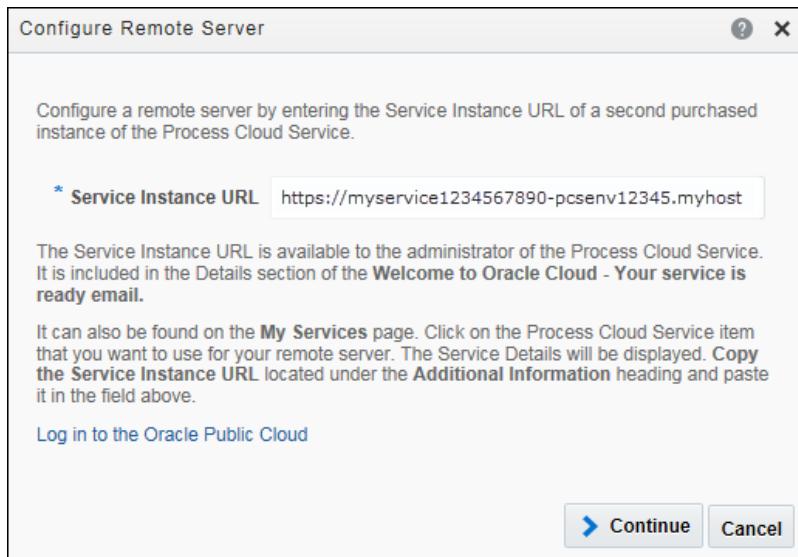
- The new service instance must be up and running before it can be configured using the design-time environment.
- Only users with administrator privileges can configure a remote server. See [Granting Administrator Privileges](#).
- You need to know the URL for your new service instance.

To configure the remote server:

1. Go to the Home page, click **Develop Processes**, and then click **Management**. The Managed Deployed Application page displays information about your available servers.



2. Click the **Click for more information** link.



3. Enter the **Service Instance URL** that points to the runtime environment. You can obtain the URL from your tenant administrator.

After the remote server is configured, it becomes available for you to manage and activate applications. The behavior of the remote environment is similar to the local environment; all options are available in both environments.

To activate an application to the remote environment, you must have privileges for that environment. Otherwise, the activation fails.

To change the URL for the remote server at any time, return to the Manage Deployed Applications page and click the **View Details** link.

Managing Security Certificates during Design Time

Certificates are used to validate external web service connections for an application when message security is applied. If an external endpoint requires a specific certificate, request the certificate and upload it into Oracle Process Cloud Service. An expired certificate results in a process instance error.

You can maintain separate certificates in design time and runtime, and you can override credentials during activation. This flexibility enables your organization to maintain separate certificates for test activation and production activation environments.

To configure message security and credentials:

1. Enable message security for a web service connector, as described in [Applying Message Security to Integrations](#).

Certificate and credential fields display when security is enabled by selecting the **APP Id - Username Token With Message Protection** option in the connector's Advanced **Security** field.

2. Specify a keystore credential and a certificate alias for the connector.
 - If needed, create a new certificate alias and upload a certificate by selecting a certificate file or pasting certificate contents.
 - If needed, create a new keystore credential and enter a name, user name, and password.

3. Manage credentials as needed. See [Configuring Credentials for Web Services](#).
4. Manage certificates as needed. You can use any of the following methods to manage certificates.

To...	Follow these steps...
Manage certificates during design time	<ol style="list-style-type: none">1. Go to Home page, click Develop Processes.2. Click Management.3. Click Options  for your server and select Manage Certificates.<ul style="list-style-type: none">• The Test tab displays information, such as alias and expiration date, for the design-time certificates uploaded in the design-time environment.• The My Server tab displays information about the run-time certificates uploaded in the runtime environment. Want to learn more about creating run-time certificates? See Managing Security Certificates during Runtime.4. Add or delete certificates.
Override certificates when you activate an application	<ol style="list-style-type: none">1. Go to Home page, click Develop Processes.2. Click Management.3. Click Options  for your server and select Deploy.4. Enter information about the application you want to activate, and then go to the Customize page.5. Use the options on the Customize page to specify services and security information.<ul style="list-style-type: none">• Select the Use design-time credentials and certificates check box to select from design-time certificates created during design time rather than runtime.• Deselect the field to display and select from run-time credentials and certificates.
Update and manage runtime certificates	<ol style="list-style-type: none">1. Go to Home page.2. Click Configure.3. Click Certificates. <p>You can create, update, and delete certificates. See Managing Security Certificates during Runtime.</p>

Part IV

Appendices

Topics

- [Basic Form and Basic Form Control Property Reference](#)
- [Basic Form Rule Examples](#)

A

Basic Form and Basic Form Control Property Reference

Look here for detailed descriptions of the properties available for basic forms and basic form controls.

Topics

- [Basic Form Properties](#)
- [Basic Form Control Properties](#)

Basic Form Properties

You can control the behavior and appearance of the basic form by changing its properties.

Properties are edited using the properties editor, which contains tabbed panes that group the basic form properties by function.

Properties on the Settings Tab for Basic Forms

These properties are available on the Settings tab when a basic form is selected in the designer.

Property	Description
Form Name	Defines the name of the form.
Description	Provides a description of the basic 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.
Element Name	Specifies the root element name in the submission XML of a form created from the basic forms designer. Must be a valid XML name. The name of the form is decoupled from the element name. If you want a form name with international characters, you must enter the international chars into the Element Name field. Note: If you change the element name, all existing submissions become invalid. The error message <i>Submission isn't valid. An incompatible change was made to the form/flow.</i> is displayed.

Property	Description
Save	Causes all submissions for this form to be stored in forms' submission repository. This property is enabled by default. If you deselect this check box, the form submission is still logged in the submission repository and you're able to view the metadata about the submission (<i>time/date submitted, success/failure conditions</i>) but no form field data is saved.
Printable	Displays a print icon at the top of your form. If you Don't want users to print your form, disable this check box. You can control which form fields are visible in the PDF print view through the printable property in each basic form control of your form.
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.
Decorated	Controls whether newly added controls have their control level decorator property set or unset. Doesn't affect controls already in the form.

Properties on the Style Tab for Basic Forms

These properties are available on the Style tab when a basic form is selected in the designer.

Property	Description
Width	Specifies the width of your form. The default "regular" width is 600px, but the drop-down also includes thin (400px) and wide (800px). The custom option enables the box to the right of the Width drop-down in which you can specify a width.
Height	Specifies an initial height for your form as it's loading into the browser. It doesn't dictate the actual height. In general, you don't have to edit the 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 the height to the actual height of your form.
Controls	Determines whether the options you define for check box and radio controls are displayed vertically or horizontally.
Layout	Select a layout. The layout changes the appearance of your form. Layouts include: <ul style="list-style-type: none"> • <i>Nouveau</i> - Takes advantage of newer browsers supporting modern CSS and is extremely well-suited to forms or flows running on mobile devices. • <i>Compact</i> - This is very similar to the Nouveau layout except for the reduction of vertical space between controls. • <i>Tight</i> - Vertical and horizontal space is dramatically reduced. Controls are positioned up against each other.
Style	Apply a style to your form by selecting a choice from the Style drop-down. Styles are mainly concerned with colors but you can also specify other properties such as font name, size and color. Choices include <i>Blue</i> , <i>Neutral</i> , <i>Green</i> or <i>Aqua</i> . You can create custom styles. Customized styles also appear in the drop-down list along with the pre-defined options.

Property	Description
Print Font	Use this property to optionally select the font used when rendering PDFs.

Basic Form Control Properties

You can control the behavior and appearance of each basic control by changing its properties.

Properties are edited using the properties editor, which contains tabbed panes grouping the properties by function.

Properties on the Settings Tab for Basic Form Controls

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 properties on the Settings tab are used by every basic form control.

Property	Description
Control Type	<p>Defines the type for this basic form control. You can change the type of the basic form control by selecting the type from the drop-down list.</p> <p>This property applies to the following basic form controls:</p> <ul style="list-style-type: none"> • Selection controls (<i>Drop-Down</i>, <i>Radio</i>, and <i>Check box</i>). • Most input controls (<i>Text</i>, <i>Text area</i>, <i>Email</i>, <i>Phone</i>, <i>Quantity</i> and <i>Number</i>). • Date controls (<i>Date</i>, <i>Time</i>, or <i>Date/Time</i>). <p>This property is populated automatically when you first add a control to a basic 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.</p> <p>You can't change a Selection control to an Input control or vice versa. You can switch a Check box to a Drop-down list, but you can't use this property to change a Check box to a Text control.</p> <p>This property is also useful for verifying what kind of controls are in your form. You assign new labels to your controls after dragging them in. This property confirms what kind of controls are in your form regardless of the labels.</p> <p>Controls generated from schema elements have a <i>Display As</i> property instead of a <i>Control Type</i> property.</p>
CSS Class	<p>Defines the controls XHTML markup class name. You can use this CSS class to reference the control in any CSS when customizing themes.</p> <p>Use the built-in CSS class name <code>f-page-break</code> to add a page break to the printed view of the forms PDF, or <code>tiff</code> by adding it to the control that should be at the top of a new page.</p>

Property	Description
Date Format	<p>Defines the date format used in the form.</p> <p>This property applies only to Date controls and the Date portion of the Date/Time control. It supports European date formatting. Dates entered into the form are translated according to the chosen format and are reformatted to match the selected format.</p> <ul style="list-style-type: none"> • A date entered into a form field is reformatted to match the selected Date Format. • A date entered into a form field is translated according to the selected format. For example, if you select a European format of DD-MM-YYYY and enter 10-05-2009, the date value is translated as May 10, 2009. If you select a US format of MM-DD-YYY the date value is translated as October 5, 2009. • If you select a format of MM-DD-YYYY using either "-" "/" or "." as the separator, either separator is valid for that format but is translated to the selected separator. • Users can still enter dates such as Feb 3, 2001. They are translated into the specified format.
Decorator	<p>If decorators are turned on for the form, when you drag an input control from the palette, it shows a default Decorator icon on the left side of the control. The default depends on the type of control being dragged in. You can change the decorator for individual controls by selecting one from the drop-down list.</p> <p>The Text Area control is the only input control that does not have a decorator</p>
Display As	<p>Lets you 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 changes the controls appearance but does not affect how the control handles data. If you must modify the controls validation behavior, you must update the schema. The Display As property is not available for the following schema controls: <i>Message, Upload, Image, Video, Link, Date, Time, Date/Time</i>.</p> <p>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).</p>
Enable if Valid	<p>Lets the user submit the form even if not all of the data has not been validated.</p> <p>By default the forms Submit button is unavailable until all required fields are filled and contain valid data. Deselect this property to allow the user to submit a form even if it's invalid.</p> <p>This property applies only to Submit controls.</p>
Enabled	<p>Determines whether the control is enabled or disabled when users first access your form. If you clear the Enabled check box, users may not enter a value in the control until the control is enabled. (You can enable the control using a Rule.)</p> <p>For example, you're creating a wedding invitation form and want to know if the people completing your form are bringing a guest. Your form includes a Text control for the guests name that becomes enabled only after users indicate (in another control) that they are coming to the wedding.</p> <p>Grouping controls cannot be disabled.</p>

Property	Description
Error Msg	<p>Specifies an error message to be displayed if the user does not supply a valid value in the control. If you leave this property blank users receive generic feedback (an "invalid value" message, for example) if they supply an invalid value.</p> <p>For example, if you're 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.</p>
Help	Specifies detailed help about a specific control. If help text is provided, an icon appears next to the control on your form. When the user clicks the icon, the help text you supplied in the Help property is displayed in a floating box.
Hide Label	Determines whether the controls label is displayed on your form. Check to hide the label on your form; clear to show the control's label.
Hint	Lets you to create a tool tip that displays in your form when the user mouses over the control.
Label	<p>Defines the label of the currently selected basic 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</p> <p>When you add a control to a basic form, it's 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 basic form.</p> <p>You can add a controls 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.</p> <p>Message controls are the only controls without labels-they are used for static text, so a label is not required in addition to the static text.</p> <p>You can use plain text or arbitrary XHMTL for the label name.</p>
Labels	<p>Lets you change the controls label. However, the XSD values for the control in the schema don't change. In other words, you can change what a user sees in the form, but not the underlying values. You can't use the [value=label] syntax you use for palette controls to change the element values.</p> <p>This property applies only to controls generated from schema elements. (If you drag a control in from the palette, it has an Options property; if the control was generated from a schema element, it has a Labels property.)</p>
Max Length	<p>Limits how many characters, expressed as a positive integer, users can supply in the control. Text area controls do not support this property due to an HTML limitation. However, it's very easy to add this functionality to the TextareaMaxLength Text area control using a business rule.</p> <p>This property is available for Text controls and other Input controls</p>
Message	Lets you include a message in your form. You can use the Rich Text Editor to format the text you enter. This property applies only to message controls and is where you enter the static text that appears on your form.

Property	Description
Message Type	<p>You can select a Message Type to display different pre-defined background colors, decorators, or a border from the Message Type drop-down on the Setting tab of the property pane. These settings can override the values you have specified for BG Color and Label Color. For these values to take effect, select <i>None</i> for the Message Type.</p> <p>Message Type values include:</p> <ul style="list-style-type: none"> • Default • None • Bordered • Success • Info • Warning • Error
Min # and Max # (Repeat)	<p>Specifies the minimum and maximum times the control can be repeated when inside a Repeat control; expressed as positive integers.</p> <p>The Min# value defines the minimum number of controls that must be added and filled-in by the end user. If the user doesn't fill in data for the minimum number of controls, the Submit button is disabled.</p>
Min # and Max #	<p>Lets you define the minimum and maximum values, expressed as positive integers, a user can input in a control.</p> <p>These properties appear as part of a Table control or when your form has an Input control inside a control and applies to the Input control. 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 are unable to submit the form. They may enter values in up to ten input controls.</p> <p>The minimum and maximum properties for the Table Control lets the user add or delete table rows in a form as required.</p> <p>Min # and Max # properties are not editable for controls generated from an uploaded schema, since the schema already specifies this through the <code>minOccurs</code> and <code>maxOccurs</code> attributes.</p>

Property	Description
Name	<p>Defines the name of the currently selected control. This name is automatically generated and defaults to the controls label less any spaces and special characters. Spaces and special characters are removed in order to make the name valid for use in rules. For XML users, this makes the name valid as an XSD schema element name. Control names are truncated to 32 characters for all controls except triggers and panels.</p> <p>If you have two controls at the same level with identical labels, the controls names are automatically made unique. If you try to edit the name such that it's no longer unique, Oracle Basic Forms prevents 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's at a different nesting level than a control dropped outside a section.</p> <p>For example, two controls, one inside a section called Car and another inside a section called Boat are also at different nesting levels. In both cases the form designer lets you name the controls identically. Both Car and Boat can contain a control named VIN.</p> <p>The Name property is used in several different contexts in Oracle Process Cloud Service:</p> <ul style="list-style-type: none"> • It determines how you reference the control within form rules. • It's used when initializing form fields through the <code>web_Forms_data</code> URL parameter for controls added from the palette. <p>Note: For controls added from XSD schema, you must use the underlying element name to initialize the control via <code>_data</code>.</p> <ul style="list-style-type: none"> • It determines how you refer to form fields from document URIs. • It's used in Form Action Display Message and Go to URL templates. • It's used in Doc Action Email Address templates. • It's the name given to the XML element corresponding to the control you drag into your form from the palette. <p>You can change the names of controls from the schema, although schema controls maintain their underlying XSD element name. For example, suppose you're using controls from two schemas in a form and both contain a control named <code>FName</code>. You can change the name of one of these controls to <code>FirstName</code> 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.</p> <p>Except for <code>_data</code>, controls from XSD, use the same rules (as above) as controls from palette.</p>
New Line	<p>The New Line property is present for all controls that have the 12 column width selector. Selecting New Line causes the control to appear on a new line in the rendered form.</p> <p>Trigger and Link controls have this property checked by default.</p>

Property	Description
Options	<p>Defines how to populate the choices the user sees in the control. This property appears for Drop-down, Radio and Check box controls.</p> <p>You may have option labels different from option values. The syntax for the options is <code><value>=<label></code>. The <code><label></code> specifies what is displayed on your form and the <code><value></code> specifies what is saved as the selected value when the user submits the form.</p> <p>When you first drag one of these controls into your form they have generic values: <i>Option 1</i>, <i>Option 2</i> and <i>Option 3</i>. This matches the generic text you see in the Properties area. To supply your own values, simply replace the values in the Options property with the values you want. Add or remove options as required.</p> <p>If you don't enter both <code><value></code> and <code><label></code> using the <code><value>=<label></code> syntax, then the value defaults to the label. As soon as you move the focus away from the Options property, options without values are automatically converted to the syntax.</p> <p>In this case the options are entered without values. In this case Oracle Basic Forms defaults the value to the label.</p> <p>The order of choices in your control matches the order in the Properties area. If you have choices that require a logical order, 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.</p> <p>In addition to the generic <i>Option 1</i>, <i>2</i> and <i>3</i> choices, a drop-down control also includes a blank option that by default appears first in the list. This blank option appears regardless of the text you supply in the Options property. You can't remove the blank option but you can make one of the other options the default.</p> <p>To specify an option string that includes the <code>"="</code> character, precede the string with <code>"a="</code>. For example, to specify the label "good = gold" specify <code>"a=good = gold"</code> for the option string.</p> <p>Choices can't be changed if they have been generated from an uploaded schema, because the schema specifies the choices. On controls generated from schema, you can't see the Options property in the Properties area. However, you can change the option Labels.</p>
Password	<p>Causes the text entered in the field to be blocked out as it's entered, in a way suitable for entry of a password.</p> <p>This property applies only to Text controls and other Input controls. If you check the <i>Password</i> check box, the text the user enters appears on the form as asterisks. However, it's submitted as normal text.</p>

Property	Description
Pattern	<p>Lets you define additional restrictions on the type of data a user inputs into a control.</p> <p>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.</p> <p>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: <code>\d{5} \d{5}-\d{4}</code>. If you type this expression in the Patterns property, your form permits values entered into this field only if they are five digits or five digits followed by the '-' character, followed by 4 digits.</p> <p>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 not type letters in a number field. Since essentially you would be attempting to expand the allowed data types in the control, Oracle Basic Forms ignores this pattern if you enter it.</p> <p>You must first save the form before the pattern takes effect. Therefore, patterns can't be tested in the form designer, only in use mode</p>
Placeholder	<p>Text entered in this field appears in your Input control until data is entered. If data is removed, the placeholder text reappears. This feature doesn't work in IE8 or IE9.</p>
Printable	<p>Determines whether or not this form field appears in the PDF document view of the form. If unchecked, the field doesn't appear in either view.</p> <p>You can set the property on a Section control and it automatically applies to all controls inside the Section control. This is often used in basic form rules.</p>
Required	<p>Specifies that a control is required. The Submit button is disabled and the form can't be submitted until all required fields contain a valid value.</p> <p>Required controls are marked with a yellow gradient.</p> <p>You can't mark grouping controls (<i>Tabs</i>, <i>Sections</i>, <i>Panels</i>, and so on) required. However, sections can be marked required. If a Section has the required property checked, required controls within it show a yellow background in design and Use mode. If the Section has required unchecked, required controls within it do not show the yellow background color in design and Use mode. The yellow background color appears only when one of the required controls in the section is filled making it mandatory to fill other required controls within the section.</p> <p>Input controls that are direct children (directly inside) of controls also can't be marked required, nor should they be, since the minimum and maximum properties define this. You may, however, make controls required when they are inside sections, tabs and panels.</p> <p>You can't edit the Required property for controls that have been generated from an uploaded schema, since the schema already specifies this through the <code>minOccurs</code> attribute. If a control from a schema appears as required and you don't want it to be required, edit the XSD and set <code>minOccurs=0</code>. Re-upload the XSD. Then the control no longer appears required.</p> <p>This property is useful when using rules to hide or show sections depending on something else in the form. If you hide the section you may have to set the Required property to false.</p>
Sensitive	<p>Controls whether the value saved in the submissions repository is hidden when viewed in the Basic Form. The value will be visible when viewed from other user interfaces such as process tracking, audit trail etc.</p>

Property	Description
Time Format	<p>Specifies the time format used by the control.</p> <p>This property applies only to the Time control created by selecting the Time or Date/Time option from the Date control drop-down. It supports standard and military time formats. Time entries entered into the form are translated according to the chosen format and are reformatted to match the selected format.</p> <ul style="list-style-type: none"> • A time entry typed into a form field is reformatted to match the selected Time Format. • A time entered into a form field is translated according to the selected format. For example, if you select a military time format of hh:mm:ss and enter 2:00 PM the time value is translated as 14:00:00.
Visible	<p>Specifies whether the control is visible or hidden when users first access your form. To dynamically show or hide the control, write a Rule that controls its visibility based on the data entered in another control.</p>
# of Rows	<p>Defines the initial size of the text area. This property applies only to Text area controls. Scroll bars appear automatically when the user reaches the # of rows specified.</p>

Properties on the Style Tab for Basic Form Controls

Use the Style tab to define display-specific properties of the currently selected control. The following table provides an alphabetical list of the properties available on the Style tab.

 **Note:**

Not all of the properties listed below are available for every basic form control.

Property	Description
BG Color	Lets you specify the color that appears behind the control. Enter any valid CSS color name or its hexadecimal RGB equivalent. For example, if you want a red background, you can type the word <i>RED</i> or <i>#aa2211</i> .
Bold	Displays the controls label in bold.
Button Color	Selects the color of a button control.
Center	Causes the content of the control to be centered. Applies to Message controls.
Date Picker	Enables a date picker control, which pops up an interactive calendar. Applies to Date controls.
Expand/Collapse	Enables an expand/collapse control. Applies to Section controls.
Italic	Displays the controls label in italics.
Item Width	Lets you change the layout of the options from vertical (one radio or check box button below the next) to horizontal. This is useful to save vertical space on long forms. It is 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 check box controls

Property	Description
Label Color	<p>Lets you 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.</p> <p>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 controls 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.</p>
Label Size	Defines the size (in pixels) of the label.
Width	<p>Used to specify the width of the control.</p> <p>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.</p> <p>All control widths are specified in columns, from a minimum of one column to a maximum of 12 columns wide. You can have any combination of controls widths on one line - (For example, three controls: 3, 4, and 5 grid columns wide respectively or four controls: 2, 2, 4, and 4 grid columns wide.).</p> <p>The width is selected by clicking on a grid in the Style tab. When you drag and drop most controls from the palette on to the canvas, the control is 12 columns wide. The trigger, link, and panel controls are the exception - the default width of the trigger and link controls is 3 columns while the panel is 6. To change the width, click the number of divisions on the grid corresponding to how wide you want the control to be.</p> <p>When you make a control n columns wide (6, for example), the entire control takes up 6 columns. As a result, if there is space, controls float up next to other controls. You can prevent this using the New Line property.</p> <p>The decorator doesn't affect the visible control width. The visible width of controls, which add up to the same width of another control, is exactly the same. For example, a control that is 2 grid columns wide + a control that is 4 grid columns wide is the same width as a control that is 6 columns wide.</p> <p>There is no Width property for section or table controls - they're always 12 columns wide. Repeats are also always 12 columns wide.</p> <p>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 is applied uniformly to each tab in group.</p> <p>When setting a controls width property you may use standard CSS-relative values; for example, 5%, 5em, 5ex or 5px.</p> <p>Note that control widths are ignored on the iPhone and the iPad even if you have explicitly specified them.</p>

B

Basic Form Rule Examples

Look here for examples of using basic form rules within a basic form.

Topics

- Calculate a Total
- Show/Hide a Billing Address
- Show/Hide a Message
- Enable/Disable a Question
- Compute Subtotals for Repeating Items
- Compute an Invoice Total
- Textarea Max Length
- Textarea Newline and Break
- Drop-Down Options
- Finding a Selected Options Index
- Synchronized Selects
- Clearing Drop-Down Options
- Default Option
- Check Box Options - Assigning Color to Check Box Choices
- Check Box Options - Making a Control Visible/Invisible Based on Check Box Choices
- Check Box Initialization
- Displaying Selected Check Box Labels
- Repeating Check Boxes
- Display a Message Control Inside a Control
- String Concatenation
- Visible/Invisible
- Visible/Invisible Section
- Select Tab
- Next Tab
- Expand/Collapse Section
- Multiple Choice
- Dynamic Options
- Triggers and Dynamic Options
- Value Change and Dynamic Options

-
- Dynamic Control Initialization
 - Verify User
 - Calculate Net Worth Using If-Else Rule
 - Dates and Times
 - Duration
 - Today's Date and Time
 - Date/Time Stamp
 - Invalid if Before Today
 - Date No More Than 14 Days From Today
 - Date No More Than 30 Days Ago
 - Update Today's Date to Central Time Zone and Adjust for Daylight Savings
 - Hours ≥ 4 and ≤ 6 Apart
 - Times
 - Show Google Maps
 - Tenants, Roles, Users
 - Item Added to Form
 - Item Added - Collapse Other Items
 - Tables
 - form.load
 - form.unload
 - Unique ID
 - Item Initialization
 - Item Added by Init Doc

For more general information about basic form rules, see [Creating Basic Form Rules](#).

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 is written as:

```
if (N1.value > 0 || N2.value > 0) {  
    T.value = N1.value + N2.value;  
}
```

This rule is automatically run whenever the user types something in *N1* or *N2* and it sets the value of *T* appropriately. You can use any legal JavaScript operators in the expression, such as subtraction or multiplication. However, it's important to make sure that the calculated value is valid in the context of type of *T*. For example, if *T* is of type integer and the computed value of the expression is decimal (such as 1.5), then the rule is attempting to set an invalid value in *T*. This generates an error. The rule sets the value as requested, marks the field as invalid, and takes appropriate action, such

as disabling the Submit button, displaying the control with a red background, and so on.

Also, if controls are added to the form from the palette, it's important to make sure they have the correct type. For example, for a numeric calculation as described above, the controls should be of type `Numeric`.

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 check box with a single option whose value is Yes. If checked, the user wants to enter a different billing address in *S* and you want to display *S*.

The form rule is written as:

```
if (B[0].value == 'Yes') {  
    S.visible = true;  
} else {  
    S.visible = false;  
}
```

This form rule is automatically run whenever the user checks or unchecks *B* and shows/hides 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 typically set the check box *B* to be initially unchecked and the section *S* to be initially hidden.

Show/Hide a Message

In the following example, create a form with a radio control named *r1*, a textArea named *t1*, and a message named *m1*. The **Visible** check box for Message *m1* is unchecked, so the message is invisible by default. The radio control *r1* has 3 buttons and the options for the buttons are:

- Option 1= New York
- Option 2= Boston
- Option 3= Albuquerque

This rule makes message *m1* visible or invisible depending on the selected options.

```
if(r1.value === 'Option 3'){  
    m1.visible=true;  
    t1.value = r1.value;  
}  
else{  
    m1.visible=false;  
    t1.value = r1.value; }
```

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 check box with a single option - Yes. If checked, the user is a smoker and you want to ask an additional question in *Q*.

The rule is written as:

```
if (B[0].value == 'Yes') {
    Q.enabled = true;
} else {
    Q.enabled = false;
}
```

This rule automatically runs whenever the user selects or deselects *B* and enables/disables the question in *Q*. Again, you can 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 typically set the check box *B* to be initially unchecked and the control *Q* to be initially disabled.

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 want to compute the subtotal.

The rule is written as:

```
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 automatically runs whenever the user enters a value in the quantity field for any item. It computes the subtotal for each item, for which the quantity is greater than 0 and fills in the subtotal field for that item with the computed value. If a particular item doesn't have a quantity, the subtotal isn't computed.

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 is written as:

```
var tot = 0;
for (var i = 0; i < S.value.length; i++) {
    tot = tot + S[i].value;
}
T.value = tot;
```

This rule runs whenever a subtotal is updated, for example, when it's updated through the rule above. It adds 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 doesn't work correctly. This is due to internal limitations in the way rules are evaluated. Note that this rule is working with controls inside a control. To handle the case of an item being deleted from the control, you require the following addition assuming that the control is named *Expense*. Table controls are *S* with a different layout. Therefore, the same applies to the table controls. If your table is named *Expense* then the control is automatically named *Expense*.

```
if (Expense.itemRemoved) { ; }
```

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 doesn't have a `maxLength` property like the text control does.

It's possible to do this using a form rule. On this control, 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 informs the user how many characters they're over the maximum allowed.

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, in an HTML formatted email, or in your form's form action display message, you must replace newline characters with HTML breaks. This is due to the fact that line breaks entered into a basic 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 rule converts the newline characters `\n` into HTML breaks.

```
var x = Description.value;
x = x.replace(/\n/g, "<br/>");
DF.value = x;
```

Drop-Down Options

The following example automatically sets the option selected in one drop-down 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 that are descriptive text. When the user selects a product, your form must perform an action based on a product ID rather than the descriptive product text. A way to do this is to have the rule that dynamically populates the product choices drop-down also populate a product ID drop-down, which remains an invisible control in the form. The product choices drop-down control is named *P* and the product ID drop-down control is named *ID*.

The first rule, *Load Products*, populates both the visible and hidden drop-downs 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];  
}
```

Sometimes the response can't be processed because the actual return element names have a dot in them. In such cases, a different syntax needs to be used that uses `[]` instead of dot-delimited. The following example shows how to access the element `"odata.metadata"` in the JSON Response using the different syntax.

```
var x;  
eval('x=' + http.get('https://localhost:7001/MasterData/api/Countries'));  
metaData.value = x["odata.metadata"];
```

Finding a Selected Options Index

The second rule *Select Product ID* keeps the hidden PID drop-down synchronized with the visible Product description drop-down.

```
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 = i;  
}
```

```
}

PID.value = PID.options[i] + '';
}
```

In v4 rules, using hidden drop-downs to keep descriptive option labels visible to the user while keeping cryptic database values hidden are often no longer necessary. Drop-down options have values distinct from the human visible option labels. The previous rule 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;
```

The following is another rule that dynamically populates both the product choices and product ID drop-downs. This rule calls a REST Service, which returns an object rather than the result set returned by the database connector, as shown previously. See [Dynamic Options](#).

```
if (S.value.length > 0) {
    eval('x=' + http.get('http://localhost:8182/products/?category=' + S.value));
    P.options = x.products;
    ID.options = x.ids;
}
```

Synchronized Selects

The previous Product Search example 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 as well as a unique product ID. The user must select a product from a drop-down on your form. You want to populate the drop-down with the product descriptions. The users don't have to see or know the product IDs, but you must use the ID as the key into the database for other selects. To do this, add another hidden drop-down to the form and populate it with the IDs. This example has a visible drop-down name *Products* and an invisible drop-down named *PID*. See the previous rule that populates these drop-downs dynamically from the database.

This rule keeps the *PID* selected option in synchronization with the selected *Product*.

```
var i;
for (x in Products.options) {
    // Determine the index of the selected product in the Products drop-down 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] + '';
```

Clearing Drop-Down Options

This sample resets a drop-down option to the automatically added blank option. For drop-downs added from palette controls and from schema, Oracle Basic Forms automatically adds a blank option so the drop-down initially shows no choice by default. To reset the drop-down, set the value of the drop-down control to null (not the

empty string). The empty string doesn't work because the empty string isn't a valid option. This rule resets the drop-down named size whenever the value of the product option changes.

```
if (product.value.length > 0) {  
    size.value = null;  
}
```

Default Option

When your options are set dynamically, as shown in the following rule, you can't set a default in the form designer. You must 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>` and not `<label> or <value>=<label>`

```
if (form.load) {  
    var cc = ['R=Red', 'B=Blue', 'G=Green'];  
    Colors.options = cc;  
    Colors.value = 'B';  
}
```

Check Box Options - Assigning Color to Check Box Choices

Check box controls are different from all other Oracle Basic Forms Palette controls in that they're multi-select. The options of the Check box form a set of name/label pairs in the following format: `option name=option label`. The value of a Check box control is an array consisting of the option names of the checked options i.e. only the selected options. Both the option names and option labels are user defined.

The following rule has a Check box control with the name `colorPalette` with these options:

- 0 = purple
- 1 = green
- 2 = blue
- 3 = yellow
- 4 = orange

The form also contains a text control with name `colorChoice`. This rule assigns `colorChoice` the choices selected from the `colorPalette` Check box. When you select an option, the name of the selected option is put into the `colorChoice` text control.

```
var checked = '';  
for (var i = 0; i < colorPalette.value.length; i++)  
{  
    checked = checked + ' ' + colorPalette.value[i];  
}  
colorChoice.value = checked;
```

When previewed, the `colorChoice` text field will update to show the numbers corresponding to the selected options, for example: 1 3

To access the labels of the checked options, the `checkbox options` property can be iterated and each value split on = to produce an array of 2 strings: the first will contain the name and the second will contain the label. Here is an example to show how the labels from the same set of options as shown above can be found:

```

var selectedColors = '';
for (var i = 0; i < colorPalette.value.length; i++)
{
    var v = colorPalette[i].value;
    for (var x in colorPalette.options) {

        var optArray = colorPalette.options[x].split("=");
        var val= optArray[0];
        var label= optArray[1];
        if (v === val) {
            selectedColors = selectedColors + ' ' + label;
        }
    }
}
colorChoice.value = selectedColors;

```

When previewed, the `colorChoice` text field will show the labels corresponding to the selected options, for example: *green yellow*.

Check Box Options - Making a Control Visible/Invisible Based on Check Box Choices

This rule makes visible/invisible a control based on the check box options a user selects. This form contains a multi-select check box named **Structures**. If the user selects the option *Detached Garage* or *House*, a text field named **Details** becomes visible.

Again, because a check box is multi-select, it's handled as an array. The array contains all selected (checked) options.

It's important to note that when a check box is added to the form from the palette and its options are multiple words containing spaces, the option array converts each space character to the `_` character. The comparison is shown below. Check box controls from schema don't have a space replaced with a `_`.

```

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 hiding **Details** also clears its value. This is because the user may have selected one of the **Structures** check boxes that made **Details** visible AND entered a value into **Details**. They may have also changed their minds and deselected 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.

Check Box Initialization

Since check box options are multi-select, in order to select multiple options using a rule, you must use the following syntax. In this example **CB** is the name of a check box control with the following options: *red, green, blue*. This rule selects all of the options.

```
CB.value = ['red', 'green', 'blue'];
```

To clear all checked options in the control named **CB**:

```
CB.value = [];
```

Displaying Selected Check Box Labels

In this example, the outcome of the rule is to display the check box labels that 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;  
        }  
    }  
}
```

Repeating Check Boxes

Check boxes inside repeat controls must be treated as an array (each check box control's values) of check box option values which is inside another array (the repeating check box control itself). This form example has a repeating section containing two controls:

- Message: Text control
- AreYouAttending: Check box 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;  
    }  
}
```

Display a Message Control Inside a Control

If you put a message control inside a control, you must add a rule to the message control to have the message when a user clicks to add a new 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 following example, the rule sets 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 (Consent.itemAdded)
{
    var index = Consent.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.';
}
```

String Concatenation

Message controls can be used in basic form rules to create summary information on your form from values entered into earlier form fields. This rule uses JavaScript variables to concatenate form field values, text strings, and HTML to format a 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/>";
```

When using field values from controls you must use a JavaScript variables and assign the concatenation to the variables, and then the variables to the Oracle Basic Forms message control value. For example, imagine you have a message control named **Summary** and a 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;
```

Visible/Invisible

This rule makes the message control `nickNameThankYou` visible when the user enters a value into the `nickName` input text control. It 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;
}
```

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; and so on. If they answer No then you want to hide all the questions specific to hearing aids. Also, when they answer Yes, you want them to answer all the hearing aid related questions.

Avoid using a 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 the patient answers Yes, you must set `HearingAid.visible=true` as well as 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` as well as all `field.required=true` again. If the **HearingAid** section contains many child controls this rule becomes very long and tedious to write.

You 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 becomes required=false.

```
if (useAid.value == 'no') {
    // Hide
    HearingAid.visible = false;
    HearingAid.required = false;
} else {
    // Show
    HearingAid.visible = true;
    HearingAid.required = true;
}
```

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 selected 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;
}
```

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;
}
```

Expand/Collapse Section

This form has three sections. The first section is expanded and the second and third are collapsed. When the user finishes their work in the first 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`. 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;
}
```

Multiple Choice

This rule makes the appropriate input text controls visible depending on the choice a user makes in a radio option control named `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;
}
```

Dynamic Options

Select control options (radios, check boxes, drop-downs, T/F) can be set dynamically using rules rather than statically using 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 shouldn't use a rule to dynamically set the options to *small*, *medium*, *large*. If you do then your form doesn't work correctly in use mode. If a user selects the option *small*, a validation error displays on the form. This is because *small* isn't one of the options allowed by your underlying XSD schema.

Triggers and Dynamic Options

This rule is executed when the user clicks the trigger controls with the name `search`. It then dynamically sets options on a drop-down list control with the 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 that contains 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 don't work in ing items.

Value Change and Dynamic Options

This rule dynamically sets the options in a drop-down 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, the series drop-down is enabled and the model drop-down is both cleared and disabled. This form contains other rules, which set the models based on the selected series. Look for the Oracle Basic 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;
}
```

Dynamic Control Initialization

This rule handles the case of setting multiple control values based on the selection of a single drop-down control. It handles this case better than using a long `if/else` construct. First add options to the drop-down named **SalesRep** in the format `<value>=<label>` where `<value>` is 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 rule then uses the drop-down value to index into the `contactInfo` 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;
```

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 `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 the **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;
    }
}

```

Calculate Net Worth Using If-Else Rule

This form contains two rules. One is adding values entered into a column of assets and a column of liabilities, and calculating `netWorth`. The second rule is checking the value of `netWorth` and displaying an error message and marking `netWorth` invalid if liabilities exceed assets. This is because the form designer doesn't want the form to be submitted 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 Basic Forms treats it exactly the same way. This is how you can dynamically control your forms' valid state.

Dates and Times

Working with dates and times is very common in most forms. The samples below show how to create the most common business logic with dates and times.

Oracle Basic Form dates can be set in the local time zone of the user by using the built-in date, time and date/time methods such as `frevvo.currentTime(form)`. Some of the samples below use the JavaScript `Date()` object. Because business rules run on the form server these dates are in the time zone where the form server is installed. There are techniques following to convert to different time zones as necessary.

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 doesn't work in a `form.load` rule unless you specify a time zone on the form's URL using the `_formTz` URL parameter. This is because the form server must know the time zone in which to return the date and time. If you don't specify a `_formTz`, the methods return null and the control values remain blank. For example, to specify Eastern time: `&_formTz=America/NewYork`.

Duration

This form initializes the hospital discharge date using a rule, and when the user enters the admission date a second 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';
    }
}
```

Today's Date and Time

Use Oracle Basic Forms' built-in date and time methods to set your date, time, and date/time controls to the current date and time in the local time zone of the user.

```
if (form.load) {
    Tm.value = frevvo.currentTime(form);
    Dt.value = frevvo.currentDate(form);
    DtTm.value = frevvo.currentDateTime(form);
}
```

The `currentTime()`, `currentDate()` and `currentDateTime()` doesn't work in a `form.load` rule unless you specify a time zone on the form's URL using the `_formTz` URL parameter. This is because the form server must know the time zone in which to return the date and time. If you don't specify a `_formTz`, the methods return null and the control values remain blank. For example `&formTz=America/New_York` sets the control values to the current date and time in the eastern time zone.

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's better to use the Oracle Basic Forms built-in date and time methods if you want to set the date to the local time zone of the user. The following method sets the date to the time zone of the form server.

```
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;
```

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;
}

```

Date No More Than 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.";
}

```

Date No More Than 30 Days Ago

This rule checks that the date entered into a control named `EventStartDate` is not more than 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;
    }
}

```

Update Today's Date to Central Time Zone and Adjust for Daylight Savings

This rule adjusts today's date from UTC time zone to Central time zone, and adjusts for daylight savings time. This additional conversion is most commonly required for

online users as the JavaScript `Date()` and Oracle Basic Forms' `DateUtil.today()` both return today's date in UTC time zone.

```
// 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 Time Zone
today.setHours(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;
}
```

Hours >= 4 and <= 6 Apart

This rule makes sure the end time is at least 4 hours greater than the start time, but no more than 6 hours later than 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;
}
}

```

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.value = ' 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;

```

Show Google Maps

Use the basic form's built in `Maps` method to create an application that displays Google Maps, by including the following text:

```

if (address.value.length > 0) {
    PanelMaps.visible = true;
    Map.value = "http://maps.google.com/?q=" + address.value;
} else {
    PanelMaps.visible = false;
    Map.value = "";
}

```

 **Note:**

You must use the message component to display Google Maps in your form.

Tenants, Roles, Users

Oracle Basic Forms has several built-in methods that lets you access information about your form server, such as the list of tenants; users in a tenant, and roles in a tenant. (Tenants allow you to segregate groups of users and roles). 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 drop-down 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 drop-down with all tenant users
eval('x=' + frevvo.listUsers(tenant.value));
Users.options = x.users;
// Init drop-down with all tenant roles
eval('x=' + frevvo.listRoles(tenant.value));
Roles.options = x.roles;
// Init drop-down 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;
    }
}

```

Item Added to Form

This rule executes when a new item is added to a form. Imagine your form contains an *ing* section named **Employee** with name `E`. Note that the name `E` is set on the control and not on the Section control. The **Employee** section control contain many controls such as *Name*, *Phone*, and so on; and a drop-down 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 Basic Forms executes this rule each time a user clicks **Add**  on the form to add a new item. Here we want to default the **Employee Shift** `ES` to the value *Day*, and populate the **Manager** drop-down dynamically with values from the Oracle Basic Forms Database Connector.

Usually your form has a `form.load` rule to initialize drop-down options for the first item visible on your form by default.

```

if (form.load)
{
    var baseURL = 'http://www.myhost.com:8080/database/';

    // Manager Drop-down

```

```

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;
    }
}

// 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 **Add**  icon you can save the overhead of going back to the database to retrieve dynamic options.

```

if (E.itemAdded)
{
    var index = E.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 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 is always `<TableName>`. For example, if you name your table `Children`, then the control is named `Children`.

Item Added - Collapse Other Items

This rule executes when a new item is added to a form. This form contains an `ing` section with a templatized label. It's efficient to collapse the other items when adding a new item to keep the list small and grid-like. `Med` is the name of the control. `Medication` is the name of the section control inside the control.

```

if (Med.itemAdded)
{
    var index = Med.itemIndex;
    for (var i = 0; i < Medication.value.length; i++) {
        if (i != index)
            Medication[i].expanded = false;
        else
            Medication[i].expanded = true;
    }
}

```

Tables

Tables are identical to controls when referenced in business rules. Tables are a grid layout of repeating items. All rule examples in this appendix that discuss `Repeats` also apply to tables.

One important note is that you can't explicitly name the control inside your table. The control inside a table is automatically named as `<TableName>`. For example, a table named `Expense` automatically has a control named `Expense`. The rule `Expense.itemAdded` and `Expense.itemIndex` references an item added to your table and that item's index respectively.

form.load

Rules can be used to initialize field values. This is a useful feature and is often used to dynamically populate drop-down 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 items added when the user clicks **Add**  , and for those added from an initial instance document. It doesn't execute for those items that you've added to your form in the Form Designer. You can either add defaults directly through the form designer or add a second rule to your form as follows.

These two rules together initialize the drop-down fields inside a control that is already in the form through the Form Designer, as well as those added each time a user clicks

Add  on the control 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 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 (E.itemAdded)
{
    var index = E.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 situation where you want to make a tab named **Review** visible only when user input matches a certain condition, in this case, the string 'TestLoad' is entered into the text control named `myText1`.

```
if (form.load) {
    myText1.value = myText2.value;
    if (myText2.value === 'TestLoad') {
        Review.visible = true;
    }
}
```

Note:

Use '`==`' for string equality in Basic Forms.

form.unload

Rules can be used to finalize field values after the users click the forms' 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 forms' order number when the form loads using `form.load`. However, if someone starts filling in the order form but never submits it, you don't want to consume the next order number in sequence if it's never 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;
}
```

Unique ID

Forms such as invoices, bills of lading, and so on often must 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 don't have to be sequential. The data named `form.id` is guaranteed to be unique for every new form instance. You can use it as follows:

```
if (form.load)
{
    InvoiceNum.value = _data.getParameter('form.id');
}
```

Item Initialization

The previous rule is 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 the value of a Repeating control. When you add a control to your form in the Form Designer you can set a default value in any of the visible Repeating controls. However, when the user

clicks **Add**  while using the form to add an additional item, the default entered in the Form Designer isn't 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 control to a default of 0. Track is the name of the control containing the input control named `albumOnly`. This rule executes each time a new Track item is added when the user clicks **Add** .

Note:

ItemAdded and itemIndex are properties of the control.

```
if (Track.itemAdded)
{
    var index = Track.itemIndex;
    albumOnly[index].value = 0;
}
```

Again, to initialize 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 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).

Item Added by Init Doc

ItemAdded also executes when Oracle Basic Forms adds items found in an init doc. You

may want to only initialize items added when the user clicks Add  on the form; not those added from an initial document. This form contains a Mailing Label that repeats. Each label requires a unique ID assigned. However, after the form is submitted the assigned IDs are saved in the database through 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 have to assign new IDs in the sequence when the user

manually adds a new Mailing Label by clicking Add  on the form. ML is the name of the Mailing Label. MLmid is the name of the ID field in the form.

```
if (ML.itemAdded)
{
    var index = ML.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);
    }
}
```