



Process Platform Entity Modeling Guide

Release 16.3

This documentation has been created for software version 16.3.

It is also valid for subsequent software versions as long as no new document version is shipped with the product or is published at <https://knowledge.opentext.com>.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111

Toll Free Canada/USA: 1-800-499-6544 | International: +800-4996-5440

Fax: +1-519-888-0677

Support: <https://support.opentext.com>

For more information, visit <https://www.opentext.com>

Copyright © 2017 Open Text. All Rights Reserved.

Trademarks owned by Open Text.

Disclaimer

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, Open Text Corporation and its affiliates accept no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Last updated: 10/31/2017

Table of Contents

Chapter 1 Introduction	13
Application development overview	14
Designing, testing, and delivering applications	15
Creating a workspace and project	17
Using the Quick Access toolbar	19
Publishing a project	19
Testing a project	20
Packaging and deploying an application	20
Process Experience	21
Chapter 2 Quick start	23
Before you begin	24
Create entities	27
Change security settings	30
Define relationships	31
Add properties	34
Create lists	39
Create forms	42
Test your application	47
Clearing the cache and refreshing the display	47
Categorizing your lists	47
Create rules	51
Take the next step	53
Chapter 3 Building the foundation	55
Entities	55
Native entities and EIS entities	55
Entity or property	56

Creating entities	57
Populating entities with building blocks	58
Building block names	61
Importing entities from database tables	63
Translating entities	67
Adding properties	67
Standard property attributes	68
Additional attributes	71
Type-specific attributes	72
Text patterns	75
Specifying an image for a property	76
Dynamic enumerations	77
Null Values	81
Validation	81
Identity	81
Adding relationships	82
Types of relationships	82
Relationship directions	83
Multiplicity	84
Relationship possibilities	84
Relationship names	86
Defining relationships	86
Deleting relationships	88
Tracking history	88
Configuring security	90
Entity permissions	90
Building block permissions	91
Assignee task permissions	94
Business workspace task permissions	94
Content task permissions	95
Discussion task permission	96
File task permissions	96
Lifecycle task permissions	97

Setting permissions	98
Conditional permissions	100
Chapter 4 Defining the process flow	111
Configuring a lifecycle	111
Case-centric solutions	111
Process-centric solutions	112
States	112
Transitions	114
Activities	116
Lifecycle examples	124
Adding a lifecycle	126
Creating a Lifecycle model	129
Configuring the task entity	137
Configuring expressions	142
Validating and building the model	143
Using the Applicability Service in a Lifecycle model	143
Adding an assignee	153
Activity flow	156
Adding an Activity flow building block	157
Automatically initiating activity flows based on conditions	158
Tracking changes	159
Chapter 5 Adding business logic	161
Adding rules	161
Rule actions	163
Configuring a rule that defines an action button	168
Using system level properties in rules	169
Using the User entity properties in rules	169
Using durations to specify relative dates and times	169
Using To Many relationships in rules	170
Using advanced functions	171
Using web services	172
Web services on entities	176
Testing and deploying entity-based web service operations	182

Web services on relationships	183
Configuring a Find web service operation	192
Web services on other building blocks	195
Entity web service details	196
SOAP web services on Process Experience	201
Setting deadlines	207
Chapter 6 Designing the presentation	211
Adding layouts	211
Item layouts	211
Creating items using an item layout	212
Home page layouts	213
Layout panels	214
Adding panels to a layout	222
Including multiple forms in a layout	225
Home page example	225
Adding a customized logo to a home page	227
Adding forms	228
Creating a form	229
Adding properties to a form	231
Adding relationships to a form	233
Adding containers to a form	246
Adding tab containers to a form	247
Adding hyperlinks to a form	248
Adding actions to a form	249
Adding images and color to a form	250
Adding text and lines to a form	252
Designing responsive forms	253
Disabling or hiding information on a form	255
Nesting forms	256
Positioning components on a form	259
Keyboard shortcuts for forms	259
Tracking form progress	260
Drop list series example	262

Adding lists	266
Configuring a list	267
Filter operators	270
Defining lists on a tasks child entity	271
Performance considerations	273
Adding a title	273
Creating an action bar	274
Grouping action buttons	275
Renaming action buttons and drop list labels	276
Adding a mobile app	277
Configuring Appworks Gateway per instance	277
Adding a Mobile App building block	278
Mobile App deployment messages	280
Mobile app panel support	280
Using Inbox lists	281
Chapter 7 Managing content	283
Adding files	284
Uploading and downloading files in Process Experience	285
Adding content	286
Adding a business workspace	288
Chapter 8 Facilitating collaboration	291
Adding a message board	291
Providing email functionality	292
Creating an email template	294
Configuring the E-mail connector	295
Creating a database configuration	295
Configuring the E-mail connector	296
Invoking the SetProfile Web service	305
Chapter 9 Sharing and re-using applications	307
Customizing an application	307
Configuring the custom application	307
Upgrading the base application	308
Updating security	309

Generating an application package for customization	309
Customizing Platform Packages	309
Importing entities from other applications	309
Managing application packages	311
Testing an application that re-uses entities from other applications	314
Chapter 10 Using the Identity package	315
Before you begin	315
Using identity-related information in Entity Modeling applications	316
Identity domain model	316
Entities	316
Address	317
Assignment	319
Country	320
County	321
EmailAddress	322
EmailAddressType	322
EmergencyContact	323
EmergencyContactRelationship	323
Enterprise	324
Genders	324
Group	325
Identity	327
Language	328
NationalId	329
Organizational unit	330
Person	333
Phone Number	338
PhoneNumberType	338
Position	339
Role	340
SocialAccount	341
SocialMedia	342
State	343

Title	343
User	344
VisaType	346
Worklist	346
Identity roles	348
OTDS	348
Information pushed from OTDS to Process Platform	348
Managing information pushed to OTDS	349
Mapping of OTDS identities to Identity entities	349
User	350
Group	351
Customizing the Identity package	351
Chapter 11 Using the expression editor	353
Dates in expressions	353
Enumerated values in expressions	353
Expression language	354
Properties	355
Decimal values	356
Special characters in string constants	357
Null values	357
Built-in functions	358
average and averageN	358
countTrue and countTrueN	358
max and maxN	359
median and medianN	359
min and minN	359
mode and modeN	359
optional	360
random	360
sum and sumN	360
isInGroup	360
isInRole	361
Operator reference	361

Date methods	365
Date methods	367
Examples	368
String methods	368
Arrays and aggregation	370
Array processing	371
Arrays and basic operators	372
Array object types	372
Aggregation operators	373
Handling uncertainty	373
The aggregation operators	374
Chapter 12 Administering an application	377
Defining security on a solution	377
Solution security policy	377
Solution creation	378
Configuring solution security	378
Defining security for Inbox Task Management	379
Properties added	380
Defining security for the Identity package	380
Defining solution variables	380
Changing the availability of a solution	381
Displaying a solution's errors and warnings	381
Configuring security for layouts	381
Deleting a solution	382
Deleting an entity	382
Publishing a solution	382
Deleting an EIS entity	383
Deleting history logs	383
Applying missing DDL statements	383
Selecting a database configuration	384
Specifying the email configuration	385
Configuring Extended ECM for Process Platform	385
Configuring a business workspace in Process Platform	385

Business Workspace Configuration Report	388
Configuring Content Server to work with business attachments in Process Platform	390
Configuring a cross application business workspace	390
Configuring a link to an existing business workspace	392
Chapter 13 EIS connectors	395
Custom EIS connectors	395
Building a custom EIS connector	395
Defining the EIS connector	396
Implementing the generated Java classes	397
Implementing the Definition Provider class	397
Implementing the Repository Provider class	407
Packaging the application	423
Using EIS entities in an application	423
Deploying the EIS connector package	423
Configuring the connection	424
Creating an EIS repository reference and importing entities	425
Building the application	425
Deploying and maintaining the application	426
Deploying the application package	426
Maintaining the connection to the repository	426
Changing the EIS connector	427
Predefined EIS connectors	428
Creating a connection to an internal system	429
Creating an MBPM connection	431
Creating a Case360 connection	436
Chapter 14 Advanced functions	441
Customizing the logo	441
Defining a custom icon for a panel header	441
Designing iHub reports on entity data	442
Publishing reports to other iHub environments	443
Displaying reports in Process Experience	444
Configuring dynamic filtering:	445
Integrating Capture Center	446

Design considerations	447
Integrating BPM processes and entities	448
Process overview	449
Creating the BPM process model	450
Creating the rule	451
Adding web services to the entity	452
Adding web services to the process	453
Creating an input message	454
Configuring multi-tenancy support	455
Creating an organization	456
Configuring an entity runtime database per organization	460
Deploying the application	461
Advanced domain modeling using subtyping	463
Replacing building blocks	465
Replacing forms	466
Replacing email	466
Conflict detection	466
Considerations for using subtyping	466
Defining security for a customization or subtype	467
Creating a theme	468
Removing a custom theme	470
Chapter 15 Tips and tricks	471
Working as a team	471
Organizing your project	472
Developing lists of master data	473
Creating lists to use with relationships in forms	474
Initiating an action when a property changes	474
Validating e-mail address properties	475
Validating a date	475

Chapter 1

Introduction

Entity Modeling is a methodology for describing the structure of business objects. It offers a compositional approach to application development. Instead of programming certain pieces of functionality repeatedly, you define the capabilities of an entity by adding functional modules (called building blocks) to it, and then configure them to suit your needs.

An entity can be anything that comprises a business requirement. Using a compositional approach, you bring many aspects needed to solve a business problem together into individual units called entities. Each entity contains properties (sometimes called fields or attributes) that hold the pieces of data to be stored for it. For example, an entity called Orders could contain properties such as Order Number, Date, Item, Color, Quantity, and so forth.

Facts about entities:

- Entities represent the things that your application interacts with.
- Entities lay the foundation for your application.
- An application's entities, and their properties and relationships, are called its domain model.
- Entities map to database tables.
- A project can contain any number of entities.

The following table gives examples of entities that could be included in various types of business applications.

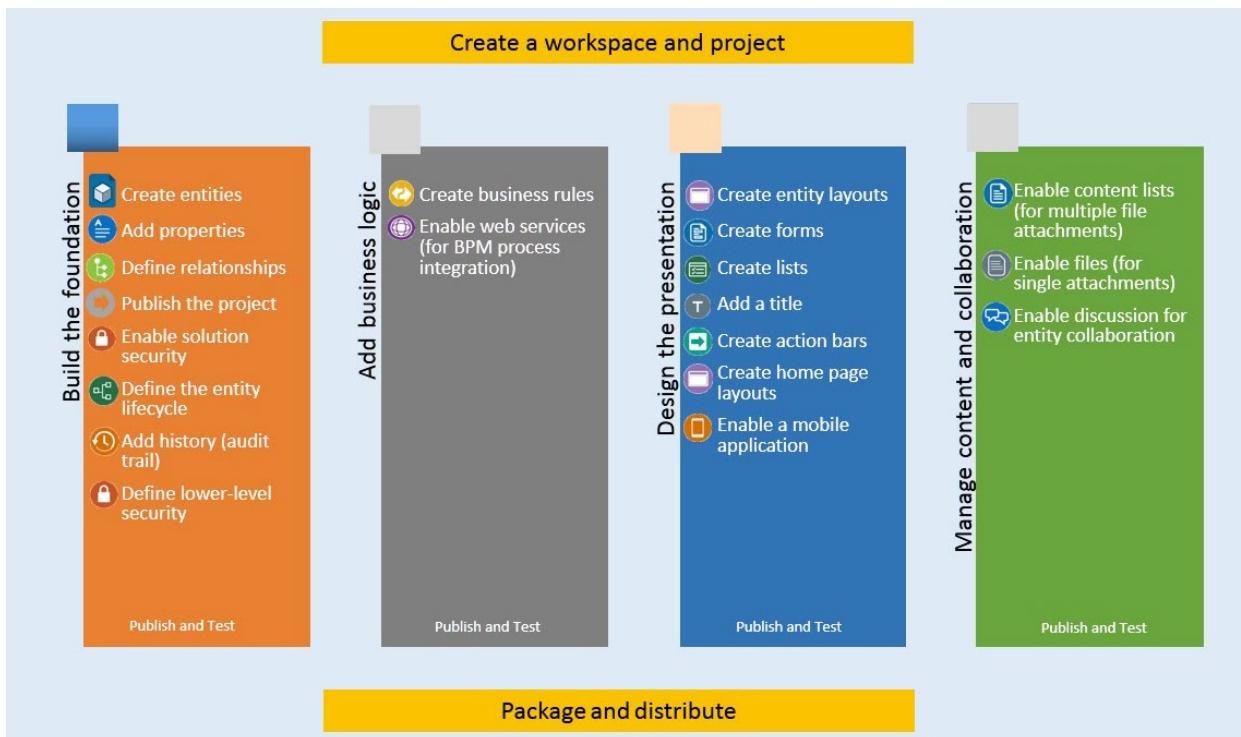
Application	Entities
Insurance	Auto Claims Policy Vehicle Auto Claim Customer
Human Resources Staffing	Job Candidate Offer Onboarding
Order Management	Order

Application	Entities
	Vendor
	Customer
	Product

Application development overview

Application development is an iterative process. The requirements of your application dictate the functions that you need. Keep in mind that not every application will include every function. The design environment provides the flexibility that you need to create an application in stages, and then build on to meet your business needs.

The following figure shows a visual depiction of the development process.



Notice that “publish and test” is shown in each section. It is important to publish and test your project often and view what users will see in Process Experience. This will also help you identify areas that you may want to change in your design.

Follow these general steps when creating an entity-based application for the first time.

1. Create a workspace and project.
2. Create entities.
3. At a minimum, create and configure the following building blocks for each entity:
 - a. Properties
 - b. Relationships

- c. Forms
 - d. Lists
 - e. Item layouts
4. Publish your project.
 5. Set solution security to enable the application for testing.
 6. View and test the application in Process Experience.
 7. Return to the design environment to continue the design process.

At this point, the way you build the application depends on your business needs. The following steps are optional and you can perform them in any order that you prefer.

1. Create home page layouts.
2. Add rules.
3. Create a BPM process and prepare for integration (via a lifecycle or web services).
4. Create an entity lifecycle.
5. Create action bars.
6. Provide users with the capability to attach content or collaborate on content. (Content, File, and Discussion)
7. Add history to maintain an audit trail.
8. Enable a mobile app based on entities.
9. Provide different levels of security access to different groups of users, so that users can perform only the operations on an entity that are granted to them.
10. Identify optional integration points (OpenText Capture Center or other external systems).

After the application design is complete, final testing can begin. When all testing is complete, the application can be packaged and deployed for production.

Designing, testing, and delivering applications

When you create an entity-based application, you follow these basic steps:

1. Create a workspace and project.
2. Create an application.
3. Publish the application.
4. Test the application.
5. Package and deploy the application to production.

While performing many of these activities, you execute actions by selecting icons. Some examples follow. See [Constructs](#) for a list of icons that are used to build a lifecycle model.

Icon	Tooltip
	Add Insert Add a User
	Align Bottom
	Align Left
	Align Right
	Align Top
	Clean Build Output (Workspace)
	Remove (Roles)
	Configure
	Configure Authentication
	Delete
	Delete selected components
	Fit to longest width
	Fit to shortest width
	Import Application Package
	Link (imported entity)
	Manage Database Configurations
	New
	Publish
	Quick Access Menu
	Refresh
	Save
	Save All
	Show/Hide Entity Properties
	Switch Solution
	Switch Workspace

Icon	Tooltip
	Synchronize All
	Test Connectivity
	Toggle grid
	Workspace Documents
	Workspace Properties Properties

Creating a workspace and project

You create an application within the Process Platform Collaborative Workspace (CWS) environment. Before you can begin, you need to create your workspace and project. A project is a container for all documents that together form an application package. It provides a single design-time view of the development content and can be validated, published, and packaged. A project can be shared by multiple solutions within the same workspace.

Important: Do not use long file names to create projects, folders, and documents because very long names interfere with the synchronization process. On the file system, the synchronized file name takes into account the entire path, starting from the installation directory including the drive letter (for example C:) so it is best practice to use as few characters as possible in file names.

You need the Developer role to create a workspace.

To create a workspace and project:

1. Start Process Platform.
2. Click the small arrow at the top of the window to "pull down" the list of applications.
3. Scroll down and select (Workspace Documents).
The Organize Workspaces window opens.
4. Click (Insert).
The Development Workspace wizard opens.
5. Provide a Name, Description, and (optional) Annotations for the workspace.
6. Select one of the following Software Configuration Management type options:

No source control - Does not associate the workspace with a Source Control Management (SCM) system.

SVN for team development - Associates the workspace with an SCM such as SVN. If you select this option, additional configuration options are displayed.

In **URL**, specify the protocol for network access of SVN. The four most common protocols that are used follow:

http://repos
https://repos
svn://repos
svn+ssh://repos

CWS supports only the first two protocols (http and https).

See the Process Platform Product Documentation for details. This option can be used to control the versions of your application or when working with multiple developers to create applications.

7. Click **Next** and type the Project Name, Package Owner, and Product Name.
The Package Owner and the Product Name are properties of the application package for delivery.

Tip: By default, the name of the project is prefaced with the name of the Package Owner in the Process Experience Administration tool. Using a unique value, such as your name, as the Package Owner will make the project easy to distinguish from other projects.
8. Click **Next** to display a summary of the information provided in the earlier screens.
9. Click **Create**.
A message appears confirming the creation of your workspace. You can click **Details** to view the workspace information in the Details tab.
10. Click **Close**.

The Workspace Documents window opens with the default view set to Explorer. However, you can change the view of Workspace Documents to Tag Search or My Recent Documents views, depending upon your development needs.

To open an existing workspace:

- In the Organize Workspaces window, double-click the name of the workspace or select it and click **Open workspace**.
The project list is displayed.

Tip: If you select **Remember my choice**, the selected workspace opens automatically when you click  (Workspace Documents).

To create a new project:

1. On the toolbar, click the down arrow next to  (New).
2. Select **Project**.
3. Type the Project Name, Package Owner, and Product Name and then click **Finish**.
The new project is created and added to the Project List.

To delete a workspace:

- In Organize Workspaces, select the check box that precedes the name of the workspace and then click  (Delete).

Using the Quick Access toolbar

A Quick Access toolbar is available in the header area of many locations within your project. Using this toolbar, you can perform common functions simply by clicking an icon. The icons that are available depend on what you are doing.

An example follows.



See [Designing, testing, and delivering applications](#) for a description of these icons and any others that may be shown on the Quick Access toolbar.

Publishing a project

By publishing your project, you make it available to users of your application. As you develop a project, publishing gives you an opportunity to catch any errors so that you can fix them before moving on to the next step. It also gives you the opportunity to verify that the application will appear to users as you intended and to make any necessary adjustments.

Tip: As you work on your application, keep CWS open in one browser window or tab and Process Experience in another. This will enable you to easily move between the design environment and the Process Experience environment.

You can publish at the project or entity level. If you are creating a new project and making many changes, you would typically publish the entire project (see [Publishing an entire project](#)). If you are making minor changes to a large stable project, you can save time by publishing the entity that you changed to detect errors before you publish the entire project (see [Publishing an entity](#)).

Important: If the changes you made affect other entities, you should finally publish the complete project to be sure to get a validated solution in Process Experience. For example, if you delete a property that is used in expressions within building blocks of other entities, publishing of the entity will succeed. However, publishing of the entire project (which also validates the other entities) is required to get a completely validated solution.

Note: When you update the definition of an entity, its internal cache (which includes a chunk of pre-allocated IDs that were assigned to it) is cleared. Therefore, you may see a jump in the ID sequence number when you publish a new property or make any other change to the entity.

By default, applications are not visible in Process Experience. Therefore, if you create entities with lists, forms, layouts, and so forth and then publish the project, you will not see your application in Process Experience unless you edit the security settings. You only need to define solution security after publishing for the first time. See [Defining security on a solution](#).

Publishing an entire project

You can publish a complete project from the CWS workspace:

To publish a project:

- In your workspace, right-click the project and select **Publish to Organization** ().

Publishing an entity

You can publish a specific entity from the CWS workspace or from the entity modeler.

- In your workspace, right-click the entity and select **Publish to Organization** ().
- In the entity modeler, click  (Publish) in the toolbar

At publishing a message appears indicating that the operation was successful or describing errors that need to be fixed. You should fix any errors before continuing development. Using the [Process Experience Administration tool](#), you may be able to see additional error information that is not exposed by the publishing process.

Testing a project

As you develop your project, you will most likely want to run your application frequently to catch any problems before you move on. Typically, you would keep both CWS and Process Experience running in separate windows to facilitate moving between the two environments.

To test your project:

1. Publish your project. See [Publishing a project](#).
2. Open Process Experience.
3. Press CTRL+F5 to clear the cache and refresh the display.
4. Test the function that is under development and any other related functions.

If you do not get the intended results, go back to CWS, make the necessary adjustments, and then publish and test again.

Packaging and deploying an application

You can deploy a business solution on a Process Platform environment by creating an application package. An application package is a bundle of documents that together form a working application. A business solution typically consists of multiple application packages, which are likely to have dependencies between them. Every project that you create in CWS corresponds to a single application package.

After you test your application, you can create and download an application package, and then deploy it to your production environment.

For detailed information on packaging and deploying your applications and solutions, see the Process Platform online help.

Process Experience

Process Experience provides an environment where authorized users work with the applications that you create. Using Entity Modeling, you customize the user interface for your application by defining a default home page for each type of user and specifying whether additional pages will be available. You also define filters that specify the items that will be available for each user to work with. Using the Security feature, you can restrict access to various parts of the application to selected users.

The *Process Experience User's Guide* provides basic instructions for using a Process Experience application.

Chapter 2

Quick start

After Process Suite is installed, you are ready to experience the simplicity of building an application using Entity Modeling. This section provides a tutorial that assumes you are a business analyst who has been asked to build an application that manages pending orders that your company has placed with outside vendors. Your application will primarily be used by the following users, although other managers and other users may also access it to view outstanding orders.

- Purchasing agents who place orders.
- Receiving area personnel who track receipt of an order.
- Administrative personnel who add new vendors to the system.

The application will include two entities (business objects) called Order and Vendor. Each entity will include the functional components (building blocks) required to create and display orders and vendors. The use case has been simplified for ease of demonstration. An actual production application would typically contain more advanced features, described in other sections of this documentation.

Note: If you prefer to work from a paper copy, you can print the Quick Start section of this guide.

Since each exercise builds on the previous one, perform them in the order in which they are presented and do not skip any. It should take about 1 to 2 hours to complete the tutorial. You can complete all the exercises in a single session or break them up into multiple sessions. However, it is best not to break up a single exercise into multiple sessions.

- [Before you begin](#)
- [Create entities](#)
- [Change security settings](#)
- [Define relationships](#)
- [Add properties](#)
- [Create lists](#)
- [Create forms](#)
- [Test your application](#)
- [Create rules](#)
- [Take the next step](#)

Tip: Deciding when to test your application is a matter of personal preference. If you prefer a cautious approach, you can test after you complete each exercise or after you complete a few exercises. However, in some cases there will be nothing to display in Process Experience. At other times, only partial functionality will be available. By publishing and testing incrementally you will be able to detect any errors that were introduced in one exercise before you proceed to the next one.

The objective of the Quick Start is to give you an understanding of the basic steps required to create an application. You can then move on to explore more advanced capabilities and customization options described in the remaining sections of this documentation. There may be some redundancy between information in the Quick Start and information in the rest of the documentation. The Quick Start is intended to expose you to the most basic features, while the rest of the documentation describes each feature in detail.

Before you begin

As you progress through the exercises, you will be working in the following environments. To facilitate moving between these environments, open each one in a different browser window.

Your system administrator can provide the URLs that you should use for each environment.

- Process Platform Collaborative Workspace (CWS) is where you will use Entity Modeling to create your application. During most of the Quick Start, you will be working in this environment.

`http://<server>:<port>/home/<organization name>`

- Process Experience is where you will test your application as a user will see it.

`http://<server>:<port>/home/<organization name>/app/processExperience/web/perform`

- The Process Experience Administration tool is where you will set security options.

`http://<server>:<port>/home/<organization name>/app/admin/web/config#`

A workspace is the starting point for application development and is the place where you maintain your projects. Before you can begin building your application, you need to create a workspace and project. To create a workspace you must have the Developer role.

To create a workspace and project in CWS:

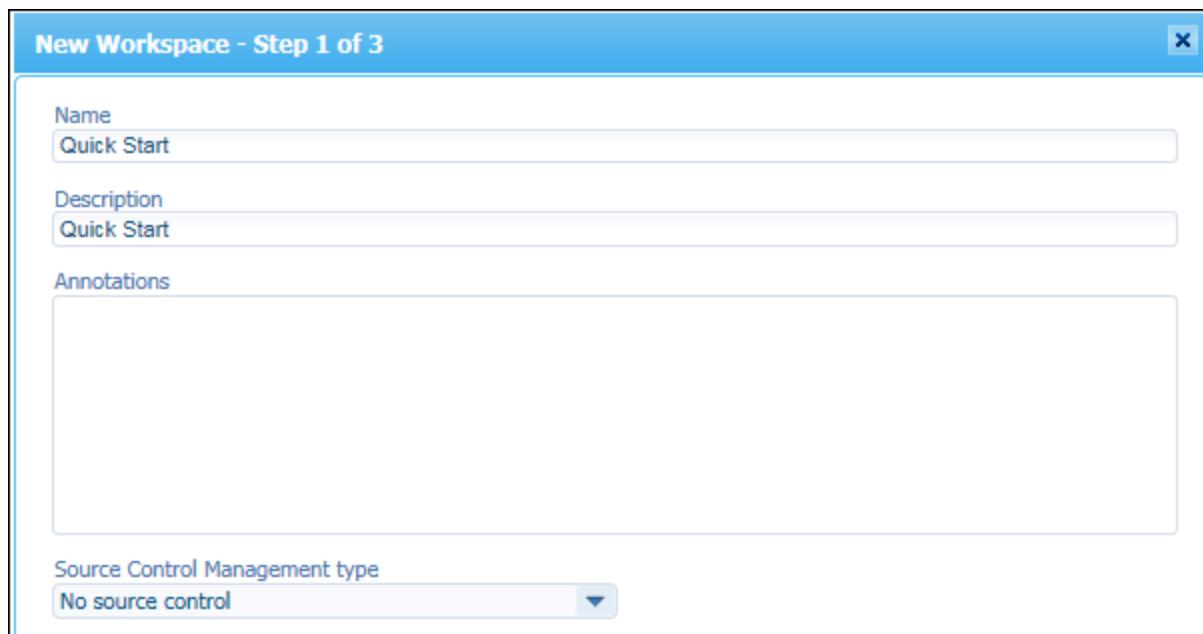
1. In Process Platform, click the small arrow at the top of the window to "pull down" the list of applications.
2. Click the down arrow at the bottom of the list of applications and select  (Workspace Documents).
The **Organize Workspaces** window opens.
3. Click  (Insert).
The **New Workspace** wizard opens.

4. In the **New Workspace - Step 1 of 3** window:

- In **Name**, type **Quick Start**.

Note: If multiple users will be running the Quick Start on the same server, include your name in the workspace name to avoid naming conflicts. For example, type **Ben Quick Start**.

- In **Description**, type **Quick Start**.
- Leave **Annotations** blank.
- In **Source Control Management type**, leave the default selection of **No source control**.



5. Click **Next**.

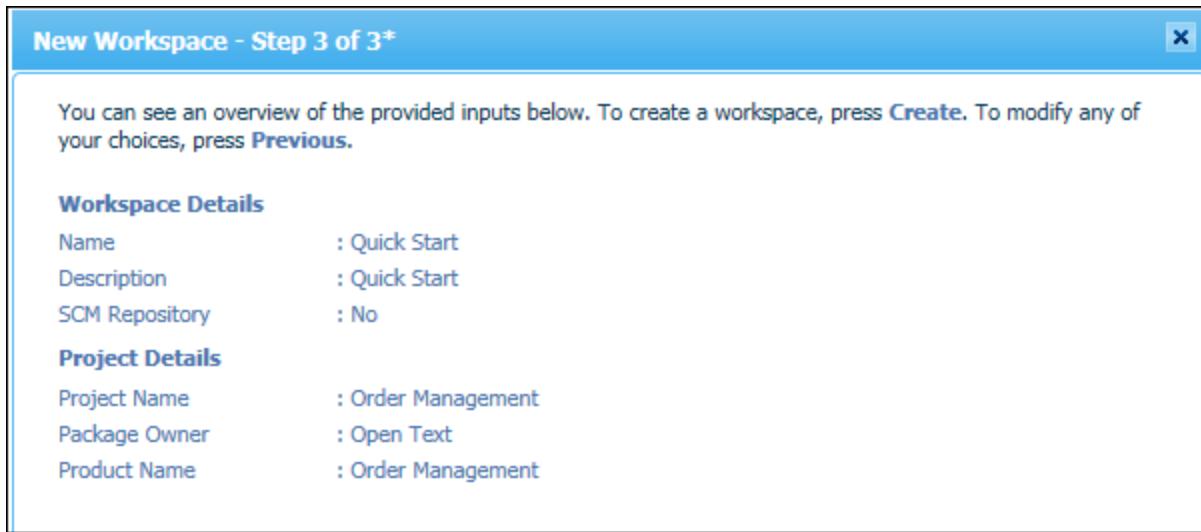
6. In the **New Workspace - Step 2 of 3** window:

- a. In **Project Name** type **Order Management**. The **Project Name** is filled in as the **Product Name**.
- b. In **Package Owner**, type **Open Text**.

Note: If multiple users will be running the Quick Start on the same server, include your name in the **Package Owner** name. For example, type **Open Text Ben**.

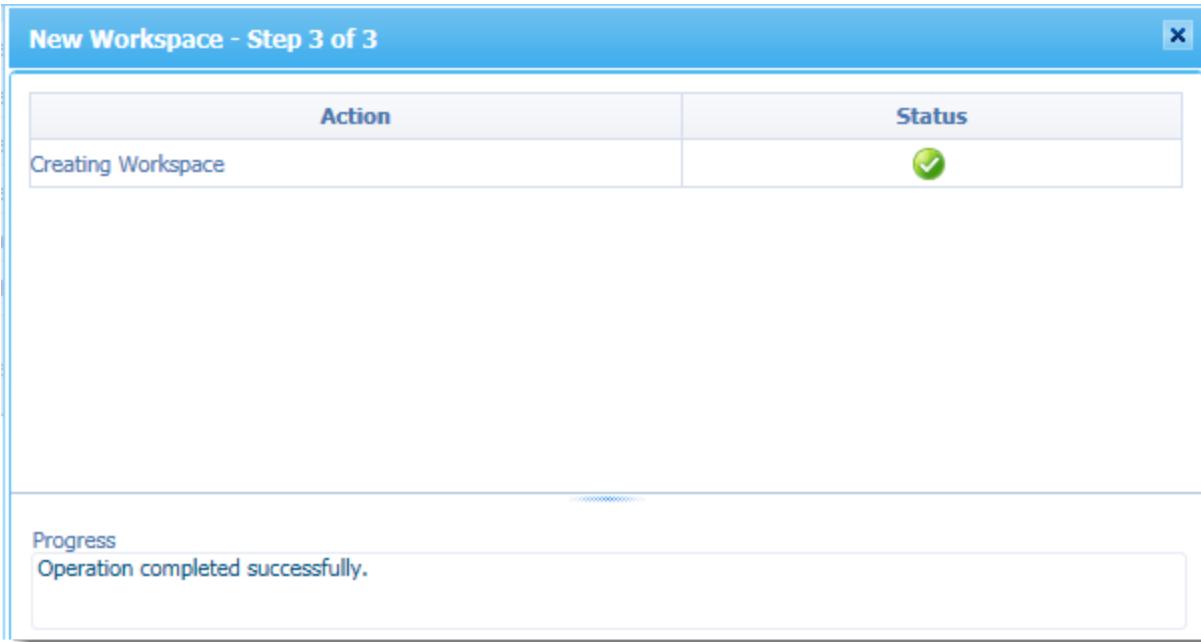


8. Click **Next**. A workspace summary is shown where you can make modifications.

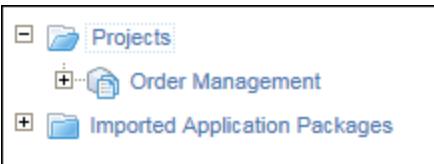


9. Click **Create**.

A message appears confirming the creation of your workspace.



- Click **Close**. Your workspace is shown in the **Workspace Documents** window. In this tutorial, you can ignore the **Imported Application Packages** folder that is added automatically.



What's next?

You are now ready to create the entities that comprise the Order Management business domain. See [Create entities](#).

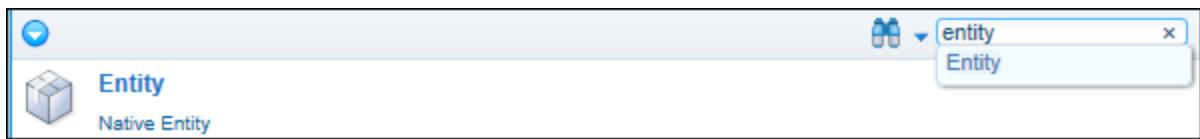
Create entities

In this exercise you will create entities (business objects) called **Order** and **Vendor**. Each entity has a Display Name and a Name.

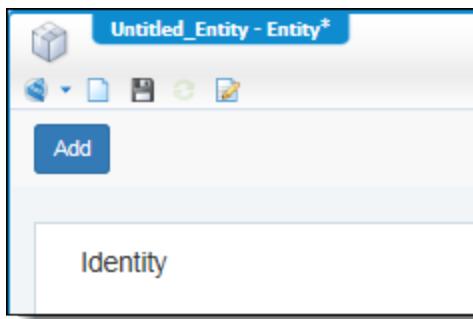
- The Display Name is used as an option for creating an order in Process Experience. For example, if the Display Name is Order, the **Create new** list includes an option called Order. If multiple users are working on the same server, include your name in the Display Name to avoid having multiple Create options with the same name. For example, type **Vendor Ben** as the Display Name.
- The Name is an internal name that cannot contain spaces. As you type the Display Name and Name, the system automatically displays suggestions. You can ignore the suggestions by typing the value you want.

To create the Order entity:

1. In **Workspace Documents**, open your workspace.
2. Click  (New).
3. Type **entity** in the search box, and then click **Entity** in the search results on the left.

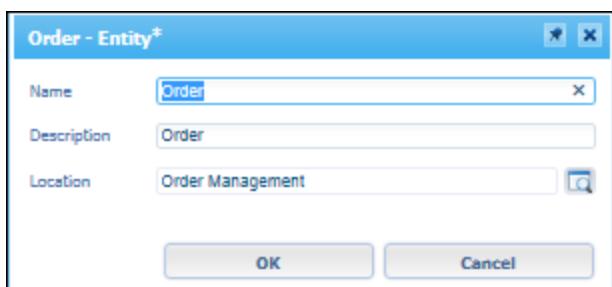


The entity is created with a name of Untitled_Entity (shown in the title bar). An Identity building block is added to the building blocks area on the left side of the window. The Identity is stored in the database with a unique ID and is used to identify the entity internally. There is no need to configure the Identity building block.



4. In the **Untitled_Entity** window, click  (Show/Hide Entity Properties).
5. In **Display Name**, type **Order <your name>**.
6. In **Name**, Type **Order**.
7. Leave all other options at their default settings.
8. Click  (Save).

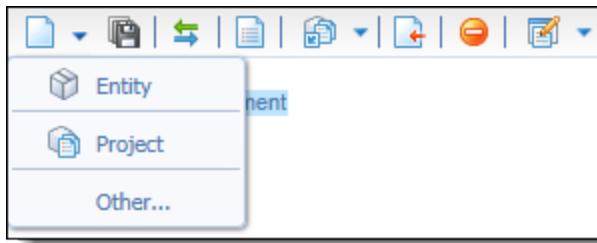
The information you entered is displayed so that you can make changes if necessary. The asterisk in the title bar indicates that the entity has not been saved.



9. Click **OK**.

To create the Vendor entity:

1. Open your workspace.
2. Click the down arrow next to  (New).
3. Select **Entity** from the recently-used list.



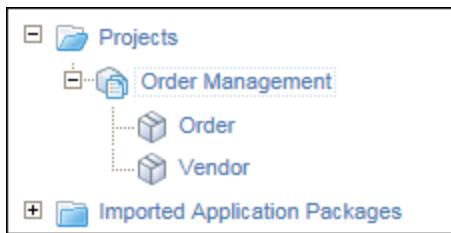
The entity is created with a name of Untitled_Entity and an Identity building block is shown in the building blocks area. The Identity is stored in the database with a unique ID and is used to identify the entity internally. There is no need to configure this building block.

4. In the **Untitled_Entity** window, click  (Show/Hide Entity Properties).
 5. In **Display Name**, type **Vendor <your name>**.
 6. In **Entity Name**, type **Vendor**.
 7. Leave all other options at their default settings.
 8. Click  (Save).
- The information you entered is displayed so that you can make changes if necessary.
9. Click **OK**.

To view your workspace and entities:

- Click the minimized **Workspace Documents** window at the bottom of the window and then expand **Order Management**.

The two entities that you created are shown in **Projects > Order Management**.



Tip: You will be accessing the **Workspace Documents**, **Order**, and **Vendor** windows frequently during application development so keep them minimized at the bottom of the window for convenience.



What's next?

You now need to publish your project to Process Experience so that you can [set the security](#) that users will need to work with it.

Change security settings

By default, applications are not visible to Process Experience users until you publish your project and configure security using the Process Experience Administration tool.

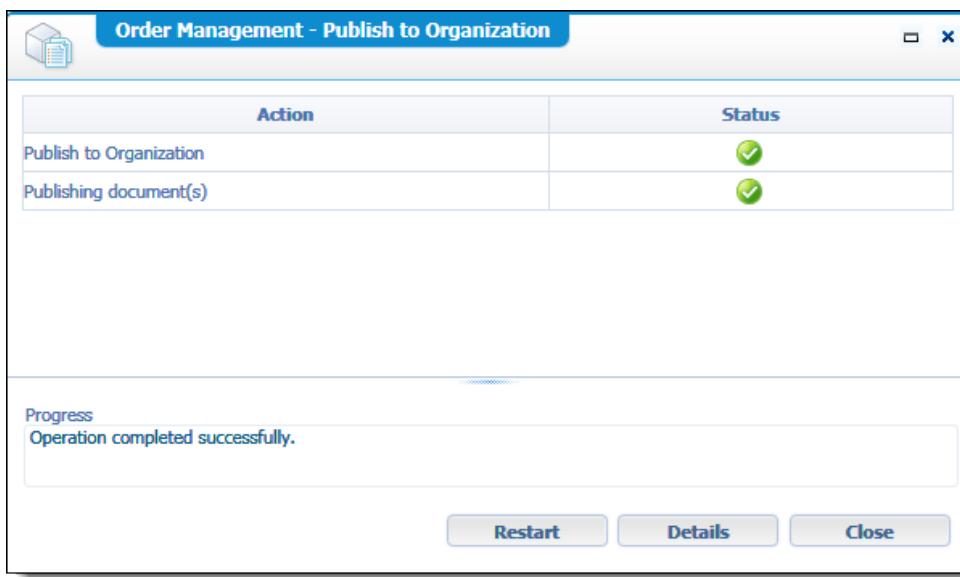
Before you begin:

- Consult your system administrator to obtain the URL for running the Process Experience Administration tool. You will use this tool to set security after you publish your project.

To publish your project:

1. In **Workspace Documents**, right-click your project and select **Publish to Organization**. or click (Publish) on the [Quick Access toolbar](#).

The following dialog box opens when the operation is complete.



2. Click **Close**.

To configure the security settings:

1. Run the Administration tool.
2. Press F5 to refresh the display.
3. In **Workspaces**, select your solution: **OpenText<your name>OrderManagement**.

Note: The workspace name is the concatenation of the name of the Package Owner and the Product Name that you entered when you created the workspace in CWS.

4. In **Configurable Elements**, expand **Solution Security** and select **Use Solution**.

5. In **Roles**, be sure that the following roles are selected:

- Entity Runtime Administrator of OpenText Entity Runtime
- Entity Runtime User of OpenText Entity Runtime
- Entity Runtime Developer of OpenText Entity Runtime

Note: Your selections are saved automatically. You do not need to click **Save** or **OK**.

With your workspace selected in the left pane, the **Configurable Elements** and **Roles** panes should look like this.

Configurable Elements	Roles
<ul style="list-style-type: none"> ➤ Solution Status ▼ Solution Security <ul style="list-style-type: none"> Administer Solution Build Include Manage Solution Manage Solution Security Use Solution 	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Entity Runtime Administrator of OpenText Entity Runtime <input checked="" type="checkbox"/> Entity Runtime User of OpenText Entity Runtime <input checked="" type="checkbox"/> Entity Runtime Developer of OpenText Entity Runtime <input type="checkbox"/> Administrator of Cordys@Work <input type="checkbox"/> Analyst of Cordys@Work <input type="checkbox"/> Audit Administrator of Cordys Audit Service <input type="checkbox"/> Audit Viewer of Cordys Audit Service <input type="checkbox"/> BAM Administrator of Cordys Business Activity Monitoring <input type="checkbox"/> BAM Analyst of Cordys Business Activity Monitoring <input type="checkbox"/> BAM Developer of Cordys Business Activity Monitoring <input type="checkbox"/> CAPAdmin of Cordys CAPConnector <input type="checkbox"/> CAPUser of Cordys CAPConnector <input type="checkbox"/> CWS Application Administrator of Cordys CWS Core <input type="checkbox"/> CWS Developer of Cordys CWS Core <input type="checkbox"/> CWS User of Cordys CWS Core

The solution is now available for every Process Experience user. When you develop actual applications that will be put into production you will need to configure more specific security policies.

What's Next?

You can now start populating your entities with the building blocks that will define their capabilities. The first step is to [define the relationships](#) between the Orders and Vendors entities.

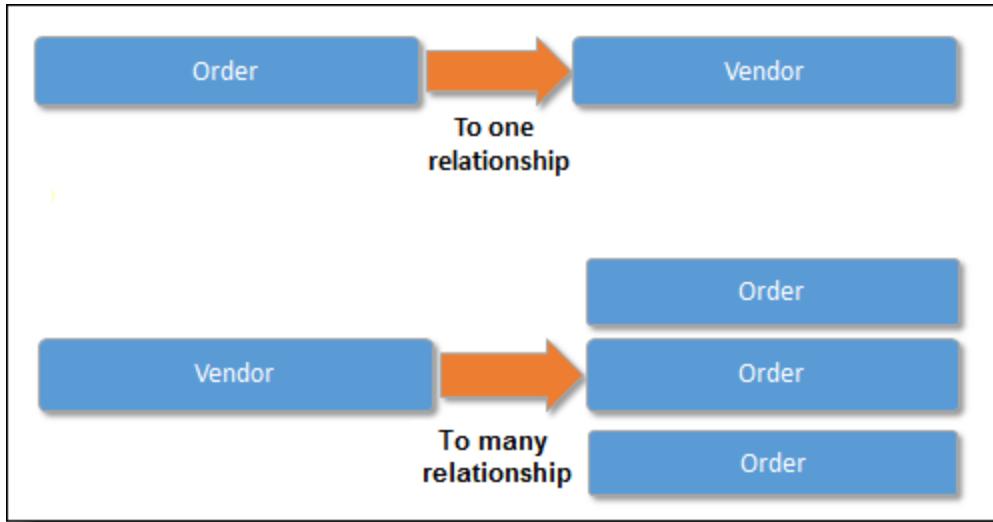
Define relationships

Entities in a business domain are often related to each other. By creating relationships between entities, business logic for one entity can depend on properties of related entities and you can include properties of related entities in forms and lists. For example, by defining a relationship from Order to Vendor, a user can select a vendor for a particular order or immediately see vendor details when opening the form.

For the Order Management application, you will create the following relationships:

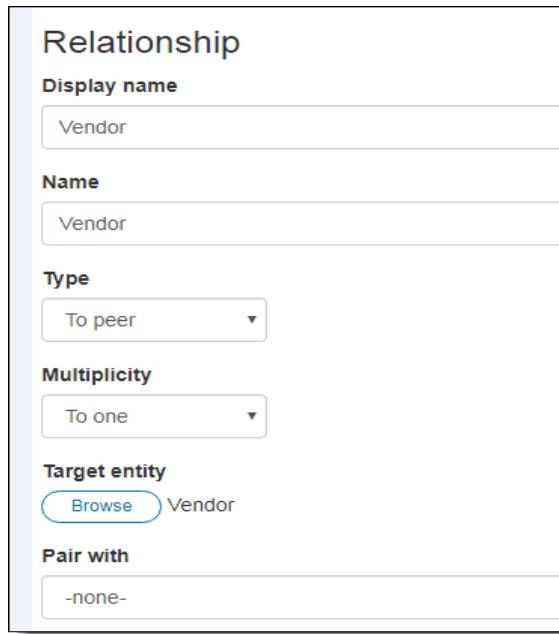
- Each order has only one vendor so you will create a To one relationship from Order to Vendor.
- Each vendor can have multiple orders so you will create a To many relationship from Vendor to Order.

These relationships are shown in the following diagram.



To relate orders to vendors:

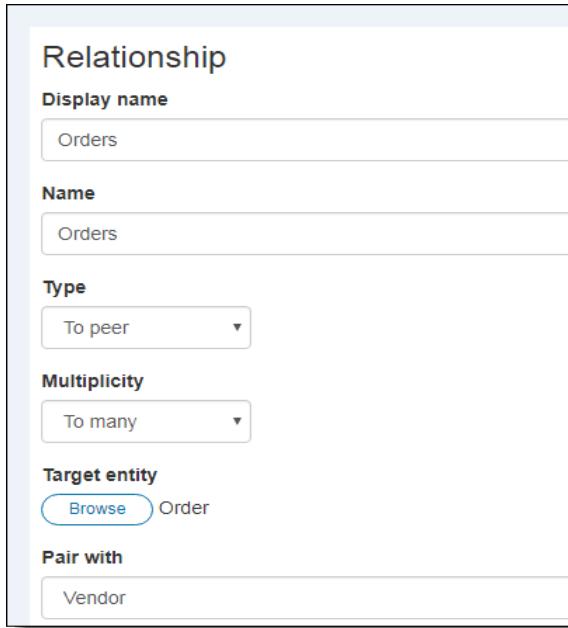
1. Open the Order entity either by double-clicking it in **Workspace Documents** or by clicking the minimized window at the bottom of the screen.
2. Click **Add > Relationship**.
3. In the **Relationship** pane, define the following options for the relationship:
 - In **Display name**, type **Vendor**.
 - In **Name**, type **Vendor**.
 - In **Type**, select **To peer**.
 - In **Multiplicity**, select **To one**.
 - In **Target entity**, click **Browse** and select **Vendor**.
 - In **Pair with**, leave the default value of -none- (no relationships are available for pairing).



4. Click **Add**.
5. Click (Save).

To relate vendors to orders:

1. Open the Vendor entity either by double-clicking it in **Workspace Documents** or by clicking the minimized window at the bottom of the screen.
2. Click **Add > Relationship**.
3. In the **Relationship** pane, define the following options for the relationship:
 - In **Display name**, type **Orders**.
 - In **Name**, type **Orders**.
 - In **Type**, select **To peer**.
 - In **Multiplicity**, select **To many**.
 - In **Target entity**, click **Browse** and select **Order**.
 - In **Pair with**, select **Vendor**.



4. Click **Add**.
5. Click (Save).
6. To check your work, open **Workspace Documents**, right-click the Order Management project, and select **Validate**.

What's Next?

The next step is to [add properties](#) to the entities.

Add properties

You are now ready to add properties to the Order and Vendor entities. Properties are the details (sometimes called attributes or fields) associated with an entity. For example, each order contains details such as order number, order date, item, quantity, and color. You will configure the Color property so that a user of your application will have a predefined list of values to select from when creating an order.

Note: In a production application, **Item** would most likely be a separate related entity that contains a list of items. In this Quick Start, item is defined as a text property for simplicity.

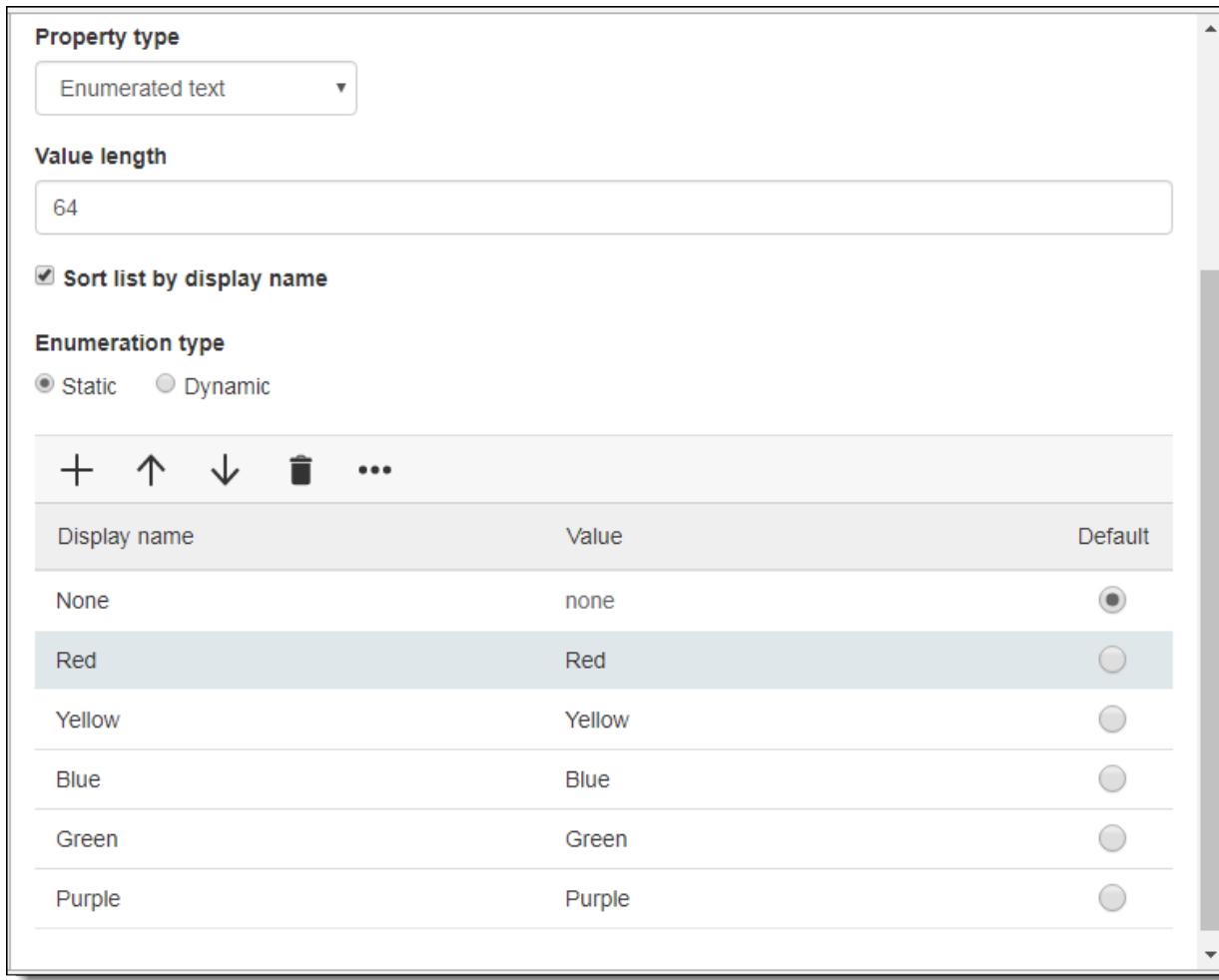
Properties are often used in forms to display the value to end users, in rules to specify business logic, or in lists.

To add properties for the Order entity:

1. Open the **Order** entity.
2. Click **Add > Property**.
3. In the **Property Details** pane, add the following properties (do not change any of the other values).
Click **Add and continue** (at the bottom of the window) after adding each property.

Display name	Name	Property type
Order Number	OrderNumber	Text
Order Date	OrderDate	Date
Item	Item	Text
Quantity	Quantity	Text

4. Add one more property called **Color** and configure it as follows:
- a. In **Display name** and **Name**, type **Color**.
 - b. In **Property type**, select **Enumerated text**.
 - c. In **Value length**, leave the default value.
 - d. Select **Sort list by display name**.
 - e. In **Enumeration type**, select **Static**.
 - f. In the **Display name** column for **none**, type **None** and then press Enter or click **+** (Add).
 - g. In the **Display name** column of the new row, type **Red** and then press Tab.
 - h. In the **Value** column of the new row, type **Red** and then press Enter (or click **+** Add).
 - i. Continue adding the following colors: **Yellow, Blue, Green, Purple**.
The list should look as follows.



5. Click **Add** (at the bottom of the window).
6. Click (Save).
7. To verify your work, open **Workspace Documents**, right-click the Order Management project, and select **Validate**.

When you open the Order entity, the properties that you added are shown in the Properties section of the building blocks list. If you click a property, additional options are displayed (depending on the property type). You will not change these options in this exercise.

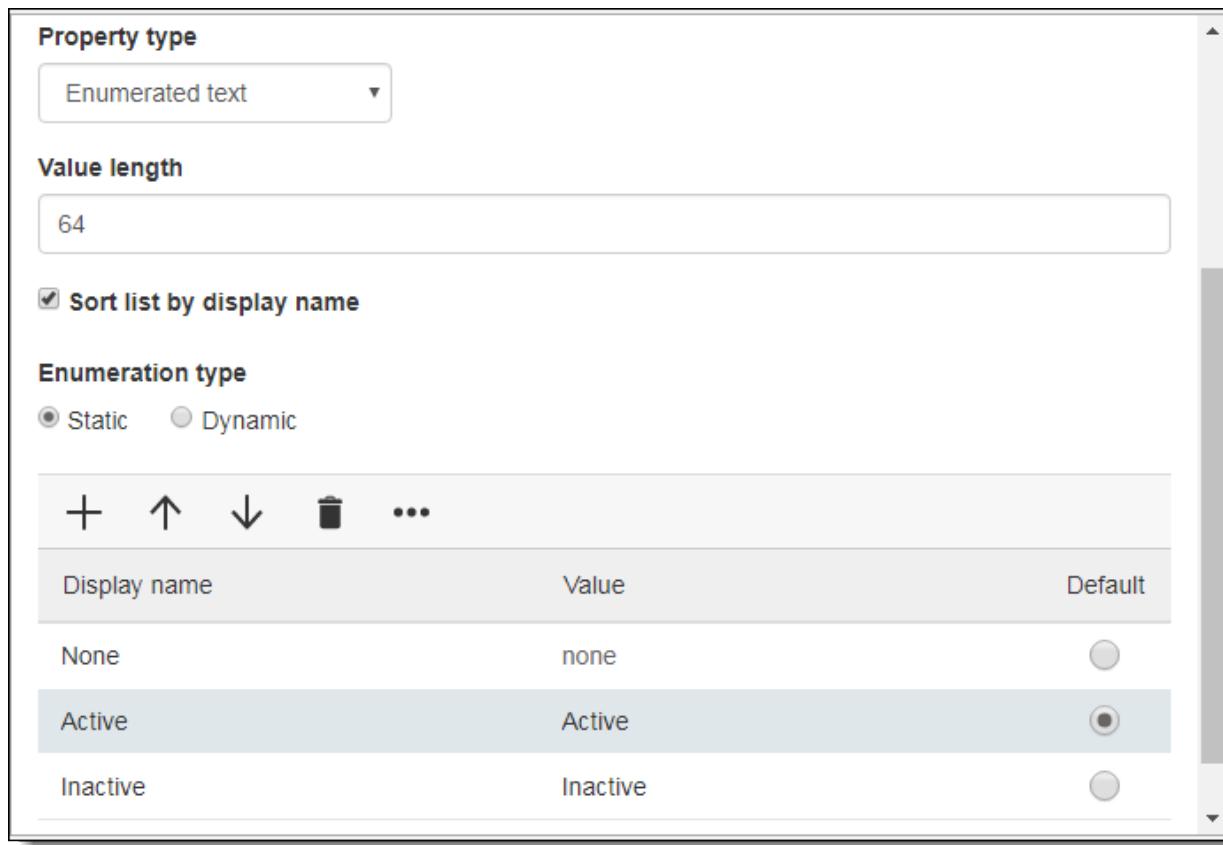
To add properties for the Vendor entity:

1. Open the **Vendor** entity.
2. Click **Add > Property**.
3. Add the following properties (do not change the default length).
Click **Add and continue** (at the bottom of the window) after adding each property.

Display name	Name	Property type
Vendor Name	VendorName	Text
Vendor ID	VendorID	Text
Street	Street	Text
City	City	Text
State	State	Text

4. Add one more property called **Status** and configure it as follows:
- a. In **Display name** and **Name**, type **Status**.
 - b. In **Property type**, select **Enumerated text**.
 - c. In **Value length**, leave the default value.
 - d. Select **Sort list by display name**.
 - e. In **Enumeration type**, select **Static**.
 - f. In the **Display name** column for **none**, type **None** and then press Enter or click **+** (Add).
 - g. In the new row, type **Active** as the **Display name** and **Value** and then press Enter or click **+** (Add).
 - h. In the new row, type **Inactive** as the **Display name** and **Value**.
 - i. Select **Active** as the **Default value**.

The list should look as follows.



5. Click **Add** (at the bottom of the window).
6. Click (Save).
7. To verify your work, open **Workspace Documents**, right-click the Order Management project, and select **Validate**.

For each entity, the list of building blocks should look like this.

Order building blocks	Vendor building blocks
<div style="border: 1px solid black; padding: 10px;"> Identity Properties Color Item OrderDate OrderNumber Quantity Relationships Vendor </div>	<div style="border: 1px solid black; padding: 10px;"> Identity Properties City State Status Street VendorID VendorName Relationships Orders </div>

What's Next?

Now that you have the properties you need, you can [create lists](#) to define the information that users will see in Process Experience.

Create lists

A list is used in Process Experience to filter and display a set of items based on your configuration. The list consists of columns, one for each property that you select to be shown. You also specify how the columns are configured and how the data will be presented. Lists are the most common way for a user to interact with the system. The user opens a list of items, then opens an item from the list or immediately performs an action on it.

In this exercise, you will create a list for the Order entity and one for the Vendor entity. You will reference these lists when you later design forms to list orders and vendors.

In Process Experience, the Display Name that you enter is shown as the name of the list in Process Experience. If multiple users are working on the same server, include your name or other distinguishing information in the Display Name to avoid having duplicate lists with the same name. In the following procedures, **All Orders Ben** and **All Vendors Ben** are used as the Display Names.

To create a list of orders:

1. In **Workspace Documents**, open the Order entity.
2. Click **Add > List**.
3. In the **List** pane, define the following options for the list:
 - In **Display Name**, type **All Orders (<your name>)**
 - In **Name**, type **AllOrders**.
 - Leave the remaining options as they are.

List

Display Name

All Orders

Name

AllOrders

Default Category

Display Count

Never

Results per page

200

All

Visible to end user

4. Click **Add** and then click **Configure**.
5. In **Available Properties > Properties**, select the properties that Process Experience users will see in the list:
 - Color
 - Item
 - Order Date
 - Order Number
 - Quantity
6. Expand **Vendor** and, in the **Properties** list, select **Vendor Name**.
8. In the **Properties shown in results** pane, click **Collapse All** and then click **Move Up** and **Move Down** to arrange the list of properties in the order in which they will be shown as columns in Process Experience.

The screenshot shows two panes side-by-side. The left pane, titled 'Available Properties', contains a tree view of entity properties. Under 'Identity', there are four unchecked checkboxes: EntityType, Id, ItemId, and ItemStatus. Under 'Properties', there are five checked checkboxes: Color, Item, Order Date, Order Number, and Quantity. Under 'Vendor', there is a collapsed section labeled 'Identity' which contains four unchecked checkboxes: EntityType, Id, ItemId, and ItemStatus. The right pane, titled 'Properties shown in results', lists the properties selected for the results: Order Number - Text, Order Date - Date, Item - Text, Quantity - Text, Color - Enumerated Text, and Vendor Name - Text. Below this list are four buttons: Expand All, Collapse All, Remove, and Move Up, with 'Move Down' being the currently selected button.

9. Click (Save) and close the window.

To create a list of vendors:

1. In **Workspace Documents**, open the Vendor entity.
2. Click **Add > List**.
3. In the **List** pane, define the following options for the list:
 - In **Display Name**, type **All Vendors** <your name>.
 - In **Name**, type **AllVendors**.
 - Leave the remaining options as they are.
4. Click **Add** and then click **Configure**.
5. In **Available Properties > Properties**, select the properties that Process Experience users will see in the list.
6. In the **Properties shown in results** pane, click **Collapse All**.

Tip: The properties are added to the **Properties shown in results** pane in the order in which you select them. Select them in the following order so that you will not need to rearrange them.

- Vendor Name
- Vendor ID
- Status

The screenshot shows the Entity Properties dialog box. On the left, under 'Available Properties', there are two sections: 'Identity' and 'Properties'. Under 'Identity', there are four items: EntityType, Id, ItemId, and ItemStatus, all with checkboxes. Under 'Properties', there are five items: City, State, Status, Street, Vendor ID, and Vendor Name, with checkboxes. The 'Status' checkbox is checked. On the right, under 'Properties shown in results', there are three items: Vendor Name - Text, Vendor ID - Text, and Status - Enumerated Text. Below these are buttons for 'Expand All', 'Collapse All', 'Remove', 'Move Up', and 'Move Down'.

10. Click (Save) and close the window.
11. To check your work, right-click the Order Management project, and select **Validate**.

What's Next?

In the next exercise, you will [create forms](#) that will be used to add orders and vendors to the system and to display information about them.

Create forms

Users work with your application using forms that you configure based on your organization's requirements.

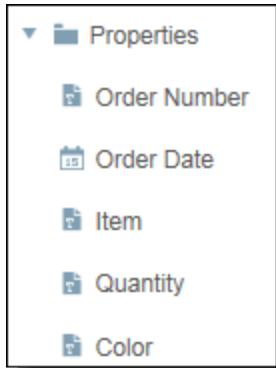
You will create each of the following forms for the Order and Vendor entities:

- Create - Process Experience users complete this form to add an order or vendor to the system.
- Default - Process Experience users use this form to view, change, or delete a previously created order or vendor.

Since you created relationships between the Order and Vendor entities earlier, you can select properties from both entities when creating forms. In the Create form of the Order entity, you will include a browse button so that users can select the vendor for the order. You will also include the related Vendor Name property to display the name of the selected vendor.

To build a form for creating orders:

1. In **Workspace Documents**, open the Order entity.
2. Click **Add > Form**.
3. In the **Form** pane, type **Create** as the **Display Name** and **Name**.
The form must be named **Create** and it must begin with an uppercase letter.
4. Click **Add** and then click **Configure**.
5. In the **Components** pane, drag the **Properties** heading for orders to the **Presentation** pane.



6. Expand the **Relationship** list and drag the **[0..1] Vendor** relationship to the **Presentation** pane.



A Browse icon (🔍) is placed in the **Presentation** area.

7. With 🔍(Browse) selected, select **Browse** and **Clear** in the **Actions** area in right pane. Process Experience users will use these icons to find and clear related vendors when creating orders.
8. Under **Browse list**, select **All Vendors**. This specifies the list that will be displayed when a Process Experience user clicks **Browse** to select a vendor.
9. In the **Components** pane, under **Relationships > [0 .. 1 Vendor > Properties**, drag the Vendor Name property to the **Presentation** area.
10. Click the **Vendor Name** property and, in **Presentation**, select **Text box**.
11. Click the **Color** property and, in **Presentation**, select **Drop list**.
12. Resize and reposition the boxes as follows.

The screenshot shows a form window with the following fields:

- Order Number: A text input field.
- Vendor Name: A text input field with a search icon (magnifying glass) and a close button (X).
- Order Date: A date input field with a calendar icon.
- Item: A text input field.
- Quantity: A text input field.
- Color: A dropdown menu currently showing "None".

13. Click (Save) and then close the window
14. To check your work, right-click the Order Management project, and select **Validate**.

To build the form for creating vendors:

1. In **Workspace Documents**, open the Vendor entity.
2. Click **Add > Form**.
3. In the **Form** pane, type **Create** as the **Display Name** and **Name**.
The form must be named **Create** and it must begin with an uppercase letter.
4. Click **Add** and then click **Configure**.
5. In the **Components** pane, drag the **Properties** heading for vendors to the **Presentation** pane.
6. Resize and reposition the boxes as follows.

Note: For **Status**, do not change the default presentation of **Radio button**.

Vendor Name	Street
<input type="text"/>	<input type="text"/>
Vendor ID	City
<input type="text"/>	<input type="text"/>
Status	State
<input type="radio"/> None <input checked="" type="radio"/> Active <input type="radio"/> Inactive	<input type="text"/>

7. Click  (Save) and then close the window.
8. Open **Workspace Documents**, right-click the Order Management project, and select **Validate**. Correct any errors that are found.

To create a default form for viewing, editing, or deleting orders:

1. In **Workspace Documents**, open the Order entity.
2. Click **Add > Form**.
3. In the **Form** pane, type **Default** as the **Display Name** and **Name**.
4. Click **Add** and then click **Configure**.
5. In the **Components** pane, expand the **Properties** list and drag the **Properties** heading for orders to the **Presentation** pane.
You can alternatively drag the properties to the form one at a time.
6. Expand the **Relationships** list and drag the **[0..1] Vendor** relationship to the **Presentation** pane.
A **Browse** () icon is placed in the **Presentation** area.
7. With  (Browse) selected, select **Browse** and **Clear** in the **Actions** area in the right pane.
Process Experience users can use these icons to change the vendor for an order.
8. Under **Browse list**, select **All Vendors** (or whatever you named the vendor list).
This list will be displayed when a Process Experience user clicks  (Browse) to find a vendor.
9. Expand **Relationships > Properties** and drag the **Vendor Name** property to the **Presentation** pane.
10. Select the **Color** property and, in the right pane, change the value in the **Presentation** list to **Drop list**.
11. Resize and reposition the boxes as follows:

The screenshot shows a Microsoft Dynamics 365 form window titled 'Order Management'. It contains several input fields: 'Order Number' (text box), 'Quantity' (text box), 'Order Date' (text box with a small calendar icon to its right), 'Color' (dropdown menu showing 'None'), 'Item' (text box), and 'Vendor Name' (text box with a magnifying glass icon for search and a 'X' icon for clear). At the bottom right of the form area are 'Save' and 'Close' buttons.

12. Click (Save) and then close the window.
13. To verify your work, right-click Order Management and select **Validate**.

To create a default form for viewing, editing, or deleting vendors:

1. Open the Vendor entity.
2. Click **Add > Form**.
3. In the **Form Details** pane, type **Default** as both the **Display Name** and **Name**.
4. Click **Add** and then click **Configure**.
5. Drag the following properties to the **Presentation** pane and resize and reposition them as shown.

The screenshot shows a Microsoft Dynamics 365 form window titled 'Vendor'. It contains fields for 'Vendor Name' and 'Street' (both in text boxes), 'Vendor ID' and 'City' (both in text boxes), 'Status' (dropdown menu showing 'Active') and 'State' (text box). At the bottom right of the form area are 'Save' and 'Close' buttons.

6. Select **Status** and change the **Presentation > Type** value in the right pane to **Drop list**.
7. Click (Save) and then close the window.
8. Publish your project by right-clicking **Order Management** and selecting **Publish to organization**.

What's Next?

It is now time to [test your application](#) so that you can use your forms to create vendors and orders. You will then access the lists you created to list the vendors and forms and to display details about a specific vendor or form.

Test your application

At this point, you are ready to test your application. To get started, run Process Experience.

When Process Experience starts, your default home page opens. You can select Home Page from the drop-down list in the header area at any time to return to where you started.

Clearing the cache and refreshing the display

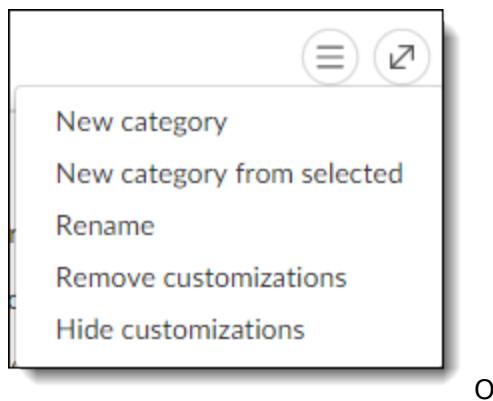
To clear the cache and refresh the display so that you can see your recently-published changes, be sure to press CTRL+F5 when Process Experience starts.

Categorizing your lists

Although it is not a requirement, it can be very helpful to organize your lists into one or more categories to make lists from the same application available from a single location. This is especially useful if there are many lists or similarly-named lists in the Lists panel. Otherwise the lists are listed in alphabetical order.

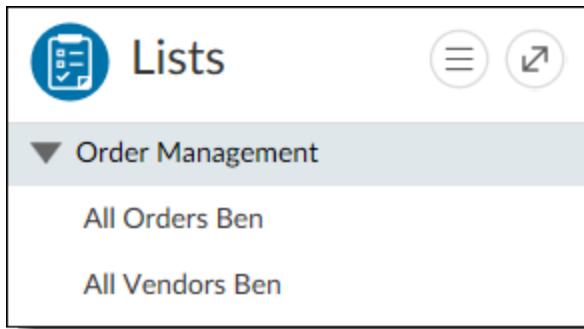
To create a category for your lists:

1. In the **Lists** panel, click the **Actions** menu and select **New category**.



2. In the text box that opens, type your name (or other text to identify the category).
3. Click (Save) at the right side of the text box.
The category you created is shown in the **Lists** panel, where you can expand or collapse it.

4. Drag the **All Orders** and **All Vendors** lists to the category you created.
Since you have not created any orders or vendors, if you click a list you will see only the column headings that you configured.



Although you will be including additional functionality in the final two exercises, this is a good time to test what you have done so far.

Perform the following procedures in the order in which they are presented.

To create vendors:

1. Click **+** (Create a new item) in the header area and then select **Vendor** (<your name>).

A screenshot of a vendor creation form. It has two columns. The left column contains fields for "Vendor Name" (with a text input field), "Vendor ID" (with a text input field), and "Status" (with radio buttons for "None", "Active" (which is selected), and "Inactive"). The right column contains fields for "Street" (with a text input field), "City" (with a text input field), and "State" (with a text input field).

2. Enter information about the vendor and then click **Save and create another** at the bottom of the window.
3. Continue creating vendors until you have about 4 or 5 of them.
(Click **Create** instead of **Save and create another** after you create the final vendor.)

To create orders:

1. Click **+** (Create a new item) in the header area and then select **Order**.

The screenshot shows a user interface for creating an order. At the top, there are two input fields: 'Order Number' and 'Vendor Name'. Below these are four more fields: 'Order Date' (with a calendar icon), 'Item', 'Quantity', and 'Color' (with a dropdown menu showing 'None'). To the right of the 'Vendor Name' field is a search icon (magnifying glass) and a close button (X).

2. Enter information about the order in the **Order Number**, **Order Date**, **Item**, **Quantity**, and **Color** fields.
3. Click  (Browse) next to the **Vendor Name** field.
4. Select a vendor from the list, and then click **Select**.

The screenshot shows a 'Select' dialog box with a title bar 'Select' and a close button. Below the title bar are two buttons: 'Open' and 'Delete'. The main area is a table with three columns: 'Vendor Name', 'Vendor ID', and 'Status'. There are four rows of data:

Vendor Name	Vendor ID	Status
Acme Automation	A-123	Active
Bell Industries	B-123	Active
Capital Computers	C-123	Active
Danforth Desks	D-123	Active

5. Click **Save and create another** at the bottom of the window.
6. Continue creating orders until you have about 4 or 5 of them.

To see all the vendors:

- In the **Lists** panel, select **All Vendors**. The vendors are listed in the Results panel. The properties that you selected for the list are displayed as column headings.

Tip: You can toggle sorting for a column in ascending or descending order by clicking the column heading.

All Vendors			
	Open	Delete	
<input type="checkbox"/>	Vendor Name ▲	Vendor ID	Status
<input type="checkbox"/>	Acme Automation	A-123	Active
<input type="checkbox"/>	Bell Industries	B-123	Active
<input checked="" type="checkbox"/>	Capital Computers	C-123	Active
<input type="checkbox"/>	Danforth Desks	D-123	Active

To see all the orders:

- In the **Lists** panel, select **All Orders**. The orders are listed in the **Results** panel. The properties that you selected for the list are displayed as column headings.

Tip: You can toggle sorting for a column in ascending or descending order by clicking the column heading.

All Orders					
	Open	Delete			
<input type="checkbox"/>	Order Number ▲	Order Date	Item	Quantity	Color
<input checked="" type="checkbox"/>	A-456	2/16/2016	Amp Meter	5	Red
<input type="checkbox"/>	B-456	2/16/2016	Barometer	3	Blue
<input type="checkbox"/>	C-456	2/16/2016	Computer	1	Green
<input type="checkbox"/>	D-456	2/16/2016	Desk	1	Danforth Desks

To view or edit a vendor or order:

- Select the check box for an item in the Vendor or Order Results list and click **Open**. The item details are displayed using the Default form that you created for vendors or orders. You can add, delete, or change any of the information that is shown.

To delete a vendor or order:

- Select an item in the Vendor or Order Results list and click **Delete**.

What's Next?

You will now [create a few rules](#) for handling orders.

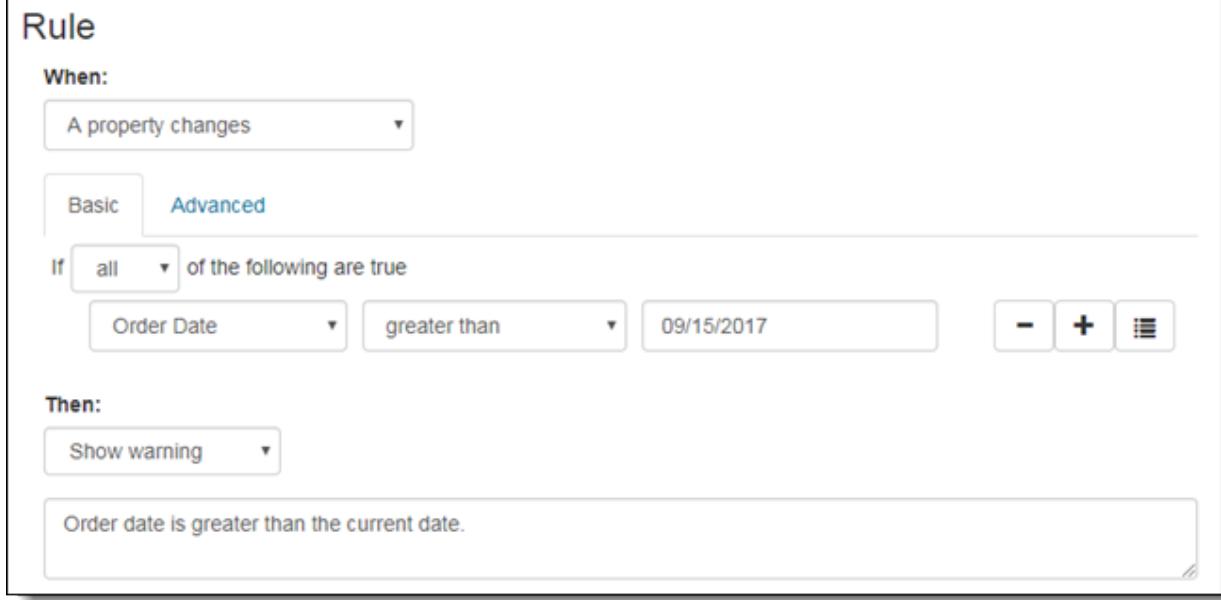
Create rules

In this exercise, you will create the following simple rules for processing orders:

- A Show warning rule that displays a warning if the order date is later than the current date. A user can ignore a Show warning rule but not a Show error rule.
- A Disable rule that disables the Color field if the item is Laptop. This will prevent users from selecting a color for a Laptop, which is available only in one color.

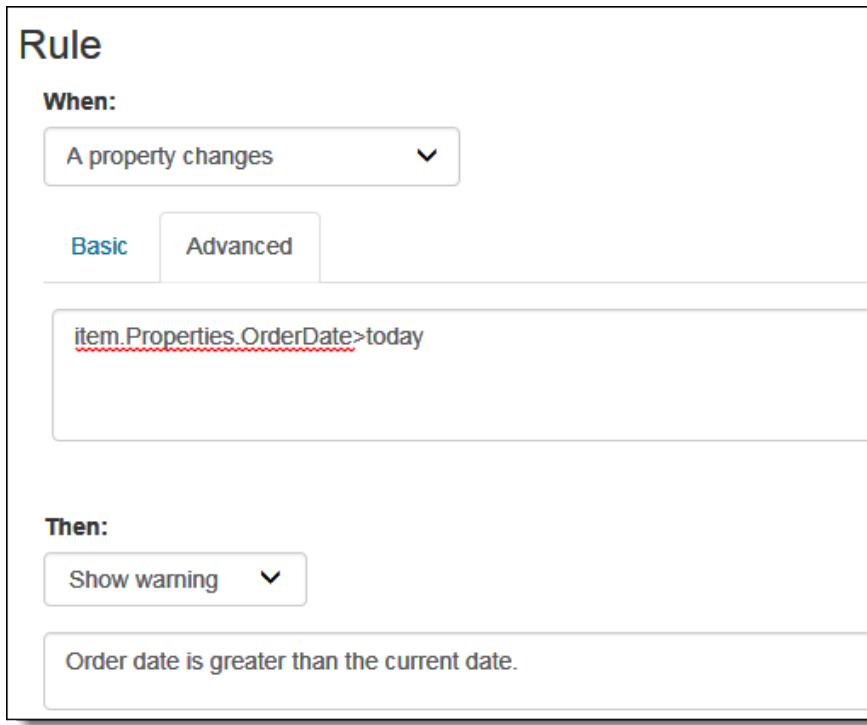
To configure the Warning rule:

1. In **Workspace Documents**, open the Order entity.
2. Click **Add > Rule**.
3. In the **Rule** pane, define the following options for the rule:
 - In **Display Name**, type **Future Date**.
 - In **Name**, type **FutureDate**.
4. Click **Add** and then click **Configure**.
5. Configure the rule as shown in the following screenshot.
To select a date, click in the value field and select Today from the calendar that appears. Notice that the current date is supplied as an absolute value.



Note: In most cases you will want the date to be relative to the current date so you will need to make the change described in the next step.

6. Click the **Advanced** tab and change the date value to
`item.Properties.OrderDate>today`



7. Click (Save) and close the window.

To configure the Disable rule:

1. Open the Order entity if it is not already open.
2. Click **Add > Rule**.
3. In the **Rule** pane, type **Disable Color** as the Display Name and **DisableColor** as the Name.
4. Click **Add** and then click **Configure**.
5. Configure the rule as follows.



6. Click (Save) and then close the window.
7. Right-click the Order Management project, and select **Publish to Organization**.

To test your rules in Process Experience:

1. Open Process Experience. Remember to press CTRL+F5 to clear the cache and refresh the window.
2. Create an order with a date later than the current date and verify that the warning is displayed as a tooltip and at the top of the window.
3. Create an order and type **Laptop** (it must start with an uppercase L) in the Item field.
4. Click outside of the **Item** field and verify that the **Color** list is disabled.

What's Next?

You are now ready to [explore](#) many additional capabilities of Entity Modeling.

Take the next step

Now that you have explored the basic features of Entity Modeling, you can expand your knowledge.

- Fine tune the Order Management application. For example,
 - Add color to forms.
 - Create a list that hides orders with a particular date range.
 - Create a simple lifecycle.
- Create a new application that solves a business problem. Start with the basics and add new functionality incrementally. For example,
 - Create an application to manage employee leave requests.
 - Create an application to manage property listings in a real estate business.

As you move on to create more sophisticated applications, see the following sections for information about additional capabilities.

[Building the foundation](#)

[Defining the process flow](#)

[Adding business logic](#)

[Designing the presentation](#)

[Managing content](#)

[Facilitating collaboration](#)

And much more

Chapter 3

Building the foundation

Entities represent the business objects that are the foundation of your application. Before you begin, create a diagram of the business objects that you need and use lines and arrows to show the relationships between them. See [Entities](#).

After you create entities, you can provide your foundation with capabilities by adding building blocks to it. See [Populating entities with building blocks](#).

You have the following options for adding building blocks to an entity:

- If you need to add only a single building block, define its attributes and then click **Add**.
- If you need to add multiple building blocks in one session, you can click **Add and continue** after you define each building block and then immediately select the next building block that you want to configure.

Entities

An entity represents a business object that is used by your organization. For example, Entities for a human resources application might be Jobs, Candidates, and so forth. Entities are typically related to each other, as when Jobs are related to Candidates. To define the business objects that an application requires, you create entities. You then create the properties that are relevant for that business object and then use those properties to create additional components such as lists, forms, layouts, rules, and others.

The entities that you configure are displayed in Process Experience so that users can create instances of these entities (entity instances are called items) and perform actions on these items as specified in the application. In Process Experience, users interact with the system (or just their application) using custom home pages, lists, preview layouts, full screen layouts, forms, and so forth.

To bring entities into your entity modeling application, you can create native entities or import entities from external sources.

Native entities and EIS entities

You use native entities to create an application whose data resides in the Process Suite database. For a native application, building blocks such as properties, lists, forms, layouts, security, history, and more are available.

You can also use Entity Modeling with existing applications whose data repositories are external to Process Suite. This is done through EIS connections that are available out-of-the-box with Process Platform. For an application on an external system, you can use the properties that exist in the external system to "decorate" the entity with lists and layouts just as if they were native entities. For example, you can create lists that filter the To Do List display for various types of users. However, since the EIS entities are not under control of Process Platform, you cannot change their structure.

Entity or property

When designing your application, it is important to consider whether each piece of information needs to be an entity or a property of an entity. In general, this depends on whether the information is specific to a particular entity or whether it will be part of a "master" list that will be related to multiple business objects (entities). For example, in the case of orders, unique information might be Order Number (each order has a unique order number). Shared information might be Vendor. For example, many orders might need to capture vendor information for creating or finding items.

The wrong approach

For this example, you might consider the following options:

- Make Vendor a text property of the Order entity.

If you do this, each user who creates an order needs to type in the vendor name. This can easily lead to inconsistencies when users type the name in different ways: Allied Corporation, Allied Corp, Allied, and so forth. Searching for multiple orders for a particular vendor will probably not provide valid results.

- Make vendor a drop-down list property of the Order entity. To create this type of property (called Enumerated Text), you can constrain the list of vendors to values that you pre-define and users must select one of those values.

This solves the problem of inconsistent data entry. However, every time you qualify a new vendor, you will need to update the list of values and republish and test the application. Additionally, the list of enumerated values would quickly become too large to be helpful.

Using either of these approaches, it is very likely that you will need to maintain multiple properties for each vendor, such as Vendor ID, Address, Discount Level, and so forth. Adding all these properties individually to each entity that requires them will quickly lead to an unwieldy list of properties.

A better approach

If you create Vendors as an entity unto itself, you can create a single "master list" of vendor information (properties) that will be available for any entity that needs it. Additionally, you can constrain the list of values that are available to users whether they create an order, a contract, or any other item where it is important to specify a vendor.

In the example presented above, a better approach is to create separate entities called Orders and Vendors, each with the properties unique to it, and create a relationship between

the entities. Using a relation to a master data entity rather than a text property in that entity avoids duplication of data which will likely become inconsistent.

If a user creating an order needs to specify a vendor, you simply create a relationship between the Orders entity and the Vendors entity and include a field on the Create Order form where the user can select a vendor name (or any other information in the vendor entity) from a list of possible values.

However, if you use a text property on Order to store the Acme vendor's name, the likely result is invoices with the vendor name set to **ACME**, **Acme**, **ACME INC**, **ACME, INC.**, and so forth. If you use a relationship and select the vendor, all of the names for the Acme vendor will all be the same and you will get much better list results.

A few tips

- It can be helpful to create a diagram of entities and their properties that shows the relationships between them. This will enable you to see whether some entities have large numbers of properties that might be better organized into separate entities.
- Over time, you will probably define a large number of relationships, so it is best practice to establish a naming convention before you start adding relationships and then use this convention consistently. In general, short names are recommended.
- Entity and property configuration is an iterative process. You will most likely need to do some fine tuning in several passes.
- It is important to consider whether you are planning to use the Lifecycle building block with an entity. Each entity can have a single Lifecycle building block. Therefore, if the items based on an entity will follow completely different Lifecycle paths, consider creating two entities, each with its own Lifecycle, and relating them if necessary.

Creating entities

The first step in creating a new application is to design the entities that model your company's business data (such as orders, customers, or products). Advance planning is critical to achieving a structure that will match your application requirements. See [Entity or property](#) for suggestions.

The Display Name of an entity is used as a **Create new** option for creating an item in Process Experience. For example, if the Display Name is Order, the **Create new** list includes an option called **Order**.

Important: In this documentation an instance of an entity is referred to as an item.

When you make changes to an entity (such as renaming or deleting), you need to publish the solution to update the relevant database tables.

Before you begin:

- Follow the instructions in [Creating a workspace and project](#).

To create an entity:

1. In **Workspace Documents**, open your workspace.
2. Click the arrow next to  (New).
3. Click **Other**, type **entity** in the search box, and then click **Entity** in the search results.

Tip: You may be able to select Entity from the recently-used list instead of searching for it.

4. Click  (Show/Hide Entity Properties) and then type a **Name** and **Display Name** for the entity.

A name can contain letters (including those from non-Latin alphabets), numbers, and underscore (_). The name cannot begin with a number. The maximum length of the name is 128 characters.

Note: If you have a large project, subtyping entities is an advanced concept designed to help make large projects more manageable. See [Advanced domain modeling using subtyping](#) for information about subtyping and about using the Abstract option.

5. If you want the entity to be available for use in other entity modeling projects, select **Available for use by others**.
6. Click  (Save).

The information you entered is displayed so that you can make any necessary changes.

7. Click **OK**. The entity is added to the project in the **Workspace Documents** (Explorer) view.

Note: The system automatically adds an Identity building block that is used internally to search for an entity by primary key. No configuration is required for the Identity building block.

8. If necessary, create additional entities.

Note: After you create at least one entity for a project you need to define security so that Process Experience users will be able to access your application. See [Defining security on a solution](#).

Deleting an entity

When you delete an entity, files that were uploaded to it are not deleted because those files may be linked to or referred to by other entities.

Populating entities with building blocks

With an understanding of entities and how they relate to your business, you can start to think about other elements that you want to build onto your entities. These elements are called building blocks.

Building blocks are categorized into the following areas:

- Basic
- Process flow
- Business logic
- Presentation
- Content and collaboration

The order in which you can add building blocks to entities is not pre-defined. This documentation is structured in a way that you would typically build an entity-based application and add building blocks to it. For example, you begin by creating entities and then adding the basic building blocks, such as properties and relationships.

You can add and configure a building block in a single operation or you can add multiple building blocks in sequence and then configure each one later.

- If you click **Add**, the building block is added to the list of building blocks for the entity and its configuration options are immediately presented.
- If you click **Add and continue**, the building block is added to the list and you can immediately add another building block. You can configure each building block later by selecting it from the list.

Basic

Basic building blocks provide the foundation for an entity.

Building block	Description
Property	Defines the pieces of information associated with entities. Properties are sometimes called attributes or fields in some systems. Property data types include text, date, integer, and so forth. See Adding properties .
Relationship	Determines how entities are connected. Typically, the entities in a business domain are related to each other. See Adding relationships .
History	Tracks specific changes or access to items to maintain a collection of history events. See Tracking history .
Security	Provides different levels of access to different groups of users so that users can perform only those operations on an entity that are granted to them. See Configuring security .
Identity	Enables synchronization of user information (identities) from Open Text Directory Services (OTDS). See Identity .

Process flow

Process flow building blocks define information about business processes.

Building block	Description
Lifecycle	Defines how an item (entity instance) evolves as the data flows through the business. See Adding a lifecycle .
Assignee	Specifies entity responsibility, including the default assignee, whether e-mail notifications are sent to assignees, and the e-mail template to use for notifications. See Adding an assignee .
Activity flow	Defines business processes based on a grid structure so that business users can manage the processes on their own without the need for IT teams. See Activity flow .
Tracking	Adds information to an entity to track when and by whom the item was created and last modified. See Tracking changes .

Business logic

Business logic building blocks define business logic and integration with other parts of Process Platform.

Building block	Description
Rule	Define business logic to specify warnings, errors, or user actions, or to trigger a business process. See Adding rules .
Web Service	Integrate your application with other systems or parts of Process Platform using web services. See Using web services .
Deadline	Specifies the schedule for completing an item. See Setting deadlines .

Presentation

Presentation building blocks define the user interface for the application.

Building block	Description
Layout	Provides customized user interfaces for your solution. A variety of layouts (item and home page) can be created that organize a set of panels into a page. See Adding layouts .
Form	Presents properties to users in customized forms. See Adding forms .
List	Filters and presents lists of items, such as work that is awaiting a user's attention. Adding lists .
Title	Displays entities with a system generated or user-specified title. See Adding a title .

Building block	Description
Action bar	Displays customized action buttons to users. See Creating an action bar .
Mobile App	Creates a mobile app based on the application. See Adding a mobile app .

Content and collaboration

Content and collaboration building blocks provide users with features to manage content and communicate about items.

Building block	Description
File	Enables users to add a single file to an item. If Content Server is used, users have additional options such as checking in and checking out versions of an item. See Adding files .
Content	Enables users to add multiple files to an item and display and delete them. See Adding content .
Business Workspace	Enables Process Experience users to perform all the document management operations supported by the Business Workspace and Folder Browser UI widgets of Content Server. See Adding a business workspace .
Discussion	Facilitates collaboration among Process Experience users. See Adding a message board .
Email	Provides email functionality in your applications. See Providing email functionality .
Email template	Creates templates that are available for Process Experience users to send emails. Creating an email template .
Inbox lists	Displays items from a user's Process Platform Inbox in a Process Experience list. Note: This feature is provided by a shared application called Inbox Task Management, which is automatically installed when you install Process Suite. See Using Inbox lists .

Building block names

Each building block has a set of general attributes such as a name, a display name, and a description. After you define the general attributes for a building block, you configure more specific attributes.

The building block name is used internally. Following are some guidelines for naming building blocks:

- The building block name cannot contain spaces or begin with a number.
- The maximum length of the name is 128 characters.
- Each building block in an entity must have a unique name, even if the building blocks are of a different kind. For example, you cannot use the same name for a form and a layout or for a list and a rule. This restriction does not apply to properties.
- All properties must have a name that is unique from other properties. That is, multiple properties cannot have the same name.
- A building block cannot have the same name as another entity in the project. For example, if your project contains an entity named "Account," you cannot also have a form named "Account" within a different entity in the project.
- For cross-database portability, property names should contain no more than 30 bytes. In English, one byte is equivalent to one character. Other languages may have different equivalencies. Consult your database administrator for details.

The following reserved words should not be used for building block names:

- Assignee
- Content
- Discussion
- DiscussionNote
- DiscussionOrganizer
- DisplayOrganization
- Email
- File
- FileContent
- FileName
- FileSize
- Group
- History
- Id
- Id1, Id2, Id3, ..., Id9
- Identity
- ItemId
- ItemId1, ItemId2, ItemId3, ..., ItemId9
- Lifecycle
- S_ITEM_STATUS
- StorageTicket
- TaskIdentity
- Title
- Tracking

Importing entities from database tables

By importing entities from existing database schemas, you can work with data from multiple applications and databases directly within Entity Modeling and Process Experience. If your organization uses separate Human Resource, Sales, and Support applications (each with its own database), you can import the data structure from each database, enhance the applications, and enable Process Experience users to work with them in a single environment. For example, for each application, you can create forms using the properties that were defined in the database and enable users to create new employee, sales, and support items using those forms. You can further enhance your applications using other building blocks.

To bring one or more legacy applications to a Process Platform environment, you import each of your existing data structures from a relational database into the Process Platform development environment. From there, you can develop your application by adding forms, rules, security, and so forth to the imported entities, publish your application to the runtime of the environment, and access it through Process Experience.

Important:

- When deploying an application package that contains imported entities, the target database for each database schema from which entities were imported must be selected individually. Be sure to pass this information along to the person in your organization who is responsible for deploying applications.
- An imported entity may or may not have an automatically generated primary key. If an imported entity does not have an automatically generated primary key you must use the Create option in grid and repeating group container controls to use a modal dialog box to create new items or the create operation will fail. See [Adding relationships to a form](#).

The Entity Importer is integrated with the Process Platform Database Metadata Modeler.

To set up Process Platform to import entities from a database:

- Add a Database Metadata document to your project and load the tables you want to import into it. See [Creating Database Metadata](#).
An imported entity is automatically linked to the underlying table and shown with  (Link).

To import data from database tables:

1. Start the **Entity Import** wizard in either of the following ways:
 - Right-click the Database Metadata document, and select **Import Entities** on the shortcut menu.
 - Open the Database Metadata document, and click **Import Entities** in the menu bar.
2. Select the tables that you want to import.
 - To select specific tables, select the check box for each table.
 - To automatically select related tables, select **Select All Related Tables**.

- To filter the list by table name, enter search text in the **Filter** box and then click **Search**. You can use the asterisk (*) as a wildcard character.
3. Click **Finish**.
- The imported entities are created in your project in the same folder as your Database Metadata document. The imported entities have the same name as the database tables from which they were imported.
- The Entity Importer maps table columns to entity properties, and table foreign keys to entity relationships. Property names are the same as column names, and relationship names are the same as the foreign key names.

Notes:

- If your database table schema changes after the initial import and you want to reflect the changes in the previously imported entities, you will need to reload the database schema into your Database Metadata document and then update the entity in the entity modeler by clicking **Update**.

Changing the data type of a primary key column is not supported. If you try to do this, an error is generated when you click **Update** and none of the schema changes are processed.

- When creating a Database Configuration using Microsoft SQL Server, the user that will be making a connection to the database must be linked to the appropriate schema. This means that you must create one user per schema and thereby one Database Configuration per user. Also, the user must not be a member of the sysadmin role.
- Tables without a primary key, or with a primary or foreign key of an unsupported data type, will not be imported as entities.
- Columns with unsupported data types are not imported as properties.
- If a column that requires a value is not imported, it is not possible to create items in Process Experience.

Mapping data types

The following table shows how SQL data types are mapped to entity property data types.

Data type	SQLServer	Oracle	MySQL	PostgreSQL	Entity property data type
BIGINT	X		X	X	Big Integer
BINARY_DOUBLE		X			Double
BINARY_FLOAT		X			Float
BIT	X				Boolean
BLOB			X		Text

Data type	SQLServer	Oracle	MySQL	PostgreSQL	Entity property data type
BOOLEAN				X	Boolean
CHAR/NCHAR	X	X	X	X	Text
CLOB/NCLOB		X			Long Text
DATE		X			Date and Time
DATE	X		X	X	Date
DATETIME	X		X		Date and Time
DECIMAL	X	X	X	X	Decimal
DOUBLE		X	X	X	Double
FLOAT(N)		X			Double
FLOAT(N)	X		X		Float for n<24 Double for 24< n <=53
INT/INTEGER	X		X	X	Integer
INT/INTEGER		X			Decimal
LONG		X			Long Text
LONGBLOB			X		Long Text
LONGTEXT			X		Long Text
MEDIUMBLOB			X		Text
MEDIUMINT			X		Integer
MEDIUMTEXT			X		Text
NUMERIC	X	X	X	X	Decimal
NUMBER		X			Decimal
REAL	X			X	Float
REAL		X	X		Double
SMALLDATETIME	X	X	X	X	Date and Time
SMALLINT	X		X	X	Integer
SMALLINT		X			Decimal
TEXT/NTEXT	X			X	Long Text
TEXT			X		Text

Data type	SQLServer	Oracle	MySQL	PostgreSQL	Entity property data type
TIMESTAMP		X	X	X	Date and Time
TINYBLOB			X		Text (255)
TINYINT	X				Integer
TINYTEXT			X		Text (255)
UNIQUEIDENTIFIER	X				Text
VARCHAR/NVARCHAR	X		X	X	Text
VARCHAR2/NVARCHAR2		X			Text
XML	X				Long Text

Linked entities

To build an application quickly from an existing set of data structures, you can import entities from existing database tables.

Initially an imported entity is shown with a  (Link) icon, indicating that it is in a linked state and no structural changes (adding/removing properties and relationships) can be made directly in the entity modeler. When an entity is linked, structural changes can only be made in its source database table and brought in by the importer.

To make structural changes in the designer you can unlink an entity from its import source by clicking **Unlink**. In this way, you do not reuse existing data but start from scratch. After an imported entity is unlinked, it behaves like a native entity and you can make any necessary changes.

Note:

- The source table for a linked entity must have a primary key defined. If you plan to use a Create form to create objects for the imported entity in Process Experience, and the primary key is not an auto-increment primary key, be sure to include the primary key property on the Create form.
- When a linked entity is a target of a To Many relationship, items cannot be created using the + (Create) action from a repeating group container or a grid unless the source table has an auto-increment primary key.
- If you unlink an entity that was imported, you cannot add a building block to it involving a child entity such as Discussion, Content, Lifecycle, or a Relationship of type Child.
- An unlinked entity cannot be linked again. For this you have to do a new import.

Translating entities

When presenting data to end users of your application, you may need to translate the name of an entity and its properties to the end user's preferred language.

Before you begin:

- Open Workspace Documents and be sure that it contains one or more entities.

To provide a translatable name for an entity:

1. In **Workspace Documents**, right-click the entity and then select **Open**.
2. Click (Show/Hide Entity Properties).
3. In **Display Name**, type the name of the entity as it should be presented to end users. As you type, the system automatically provides suggestions of translatable text that is already available in the project.
4. Use the arrow keys to navigate through the suggestions and press Enter to select one. To use different text, type it and then press Enter.
5. Click (Save).

To provide a translatable name for an entity's properties:

1. In **Workspace Documents**, open the entity.
2. Select an existing property, or add a new one.
3. In **Display Name**, type the name of the property as it should be presented to end users. As you type, the system automatically provides suggestions of translatable text that is already available in the project.
4. Use the arrow keys to navigate through the suggestions and press Enter to select one. To use different text, type it and then press Enter.
5. Click (Save).

If you provided new text, it is added to the list of suggested text for the current project.

Knowing which aspects of your application are translatable is just one part of translating an application. The most important part is about actually providing the translations for the languages you want to support. For detailed instructions on how to translate your application text, see *Translating Text to a Target Language* in the Process Platform product documentation.

Adding properties

Properties are the pieces of information (sometimes called attributes or fields) associated with an entity. When designing an application, you need to consider what you need to know about each entity and break this information down into a set of properties for it.

When Process Experience users create new items, they enter values for the properties that you configure on a form. For example, if you add properties called Name, Street, City, and

State to an Employee entity, users add values for these properties when they add a new employee to the system. They also see these properties when listing employees, viewing a particular employee record, and performing other functions.

Generally, you will add properties to an entity first so that they can be used to configure other building blocks such as forms, rules, lists, and so forth.

To add properties to an entity:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Property**.
3. In the **Property Details** pane, provide a **Display Name** and **Name**
4. In **Property Type**, select the type of data for the property.
6. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

7. If necessary, configure **Additional attributes** or **Type-specific attributes** for the property.
8. Click  (Save).

Standard property attributes

All properties have the following standard attributes that you define when adding a property to the list of building blocks.

Display Name

The display name is used as the property's default label in forms, lists, and so forth.

Name

A name can contain letters (including those from non-Latin alphabets), numbers, and underscore (_). It cannot contain a space. The name cannot begin with a number. The maximum length of the name is 128 characters. You cannot change the property name after the property is created.

Notes:

- All property names in an entity and in all of its subtyped entities throughout the entire subtype hierarchy must be unique. Deployment of an entity subtype that would introduce a duplicate property name will fail. See [Advanced domain modeling using subtyping](#).
- For cross-database portability, property names should contain no more than 30 bytes. In English, one byte is equivalent to one character. Other languages may have different equivalencies. Consult your database administrator for details.
- For a list of reserved words, see [Building block names](#).

Property Type

The property type defines the type of data the property will contain. You cannot change the property type after the property is created.

The following property types are available for adding to an entity.

Property Type	Description
Boolean	One of two possible values: true or false. A none value is created automatically. However, it is optional and you can delete it.
Currency	A currency value with a precision of 19 digits in which 4 digits are scale.
Date	A specific date.
Date and Time	A combined date and time value, accurate to 100 nanoseconds.
Decimal	A numeric value with a specified number of digits to the left and right of the decimal point.
Duration	A time span that can be used in calculations such as deadlines. A duration consists of a number and unit, such as 1 month or of multiple numbers and units, such as 1 year, 3 months, and 10 days.
Enumerated Integer	A list of integer values (with full Unicode support). When presented to a user, the list of values is presented for selection rather than prompting the user to type in a value. The first value (none) is automatically created but it is optional and you can delete it. The value and display name must be unique across all items or an error message is displayed. The maximum length of the display name is 128 characters. The maximum length of the value is the length of the property. For a dynamic enumeration, the values are retrieved using a business process that can be selected after adding the property.
Enumerated Text	A specified list of text values (with full Unicode support). When presented to a user, the list of values is presented for selection rather than prompting the user to type in a value. The first value (none) is automatically created but it is optional and you can delete it. The value and display name must be unique across all items or an error message is displayed. The maximum length of the display name is 128 characters. The maximum length of the value is the length of the property. For a dynamic enumeration, the values are retrieved using a business process that can be selected after adding the property.
Float	A 32-bit floating point numeric value with a precision of approximately up to 9 decimal digits and a range of

Property Type	Description
	approximately -10^{38} to 10^{38} . Due to their representation, floating point numbers can lose precision due to rounding and should not be used to store monetary values.
Image	An image of a supported type (JPEG, PNG, BMP, WBMP, or GIF). Note: You cannot use image properties in expressions.
Integer	A numeric value with a range of -2,147,483,648 to +2,147,483,647 with no fractional part. The thousands separator is locale specific. For example, some locales use a period as the thousands separator instead of a comma.
Long Text	A text value (with full Unicode support) with no length limit (you do not specify a length).
Text	A text value with a specified length limit.

Note: In Entity Modeling, an enumerated property or a Boolean property having no value is the same as having value 'none'. When an enumerated or boolean property is set as a required property on a form, setting the property to no value is not allowed so the value 'none' is not presented as a valid value in the radio button, dropdownlist, or listbox. When 'none' is defined as a valid value, an enumerated or boolean property will have the initial value 'none' selected on a form even if it is not explicitly set as the default value.

Important: If you publish a project after you recreate (delete and create) a property in the same entity with the same name but a different property type, the data type of the deleted property is not changed in the database. Therefore, after deleting a property that you intend to recreate you should immediately publish the project to ensure that the data type is deleted from the database.

Example:

Assume that you create a property called Sales Date and accidentally assign it a property type of Text, although you intended to assign it a property type of Date. If you delete the Sales Date property and then recreate it with the Date property type (without first publishing the project), the database is not updated with the new property type. Instead, you need to publish the project immediately after you delete the property to ensure that the Text property type is deleted from the database and then recreate the Sales Date property with the Date data type.

The following property types are available only for import.

Property Type	Description
Big Integer	A numeric value with a range of -2^{63} (-9,223,372,036,854,775,808) to $2^{63} - 1$ (9,223,372,036,854,775,807) with no fractional part. The

Property Type	Description
	thousands separator is locale specific. For example, some locales use a period as the thousands separator instead of a comma.
Double	A 64-bit floating point numeric value with a precision of approximately 16 decimal digits and a range of approximately -10^{308} to 10^{308} . Due to their representation, floating point numbers can lose precision due to rounding and should not be used to store monetary values.

Additional attributes

All properties have additional attributes that you can change (if necessary). These options are not shown while you are initially adding a property because, in many cases, you will not need to change the default values.

Tooltip

The Tooltip is shown when a Process Experience user positions the pointing device on an input field for the property. For example, when the Item number property is included on a form the tooltip might be as follows:

Enter the alphanumeric item number

Allow participants to change the value

Specifies when users can change the value of the property. The capability to update the property can be further restricted by security permissions, which are defined separately.

- Anytime - The property can be modified at any time.
- Only when first created - The property can be updated only when the object is first created.
- Never - The property cannot be modified.

When users work on this property at the same time, report conflicts

Specify how to handle conflicts that occur when multiple people change a property at the same time. A conflict is reported only if the database is updated while a form is displayed, before the current user saves changes. For example, assume that Harry and Sally display a form at the same time. When they open the form, they both see Property A with a value of 1. Harry changes the value of the property to 2. A while later, Sally changes the value of the property from 1 (which is what she still sees on her form) to 5. Without conflict detection, the value of the property is 5 (because Sally was the last person to modify the property).

With conflict detection set to **When a change is made by another user**, Sally is notified that another user made a change that conflicts with her change. The notification indicates that someone else changed the value of the property to 2. Sally is prompted about whether to proceed with her update.

Allowed values for conflict detection:

- When a change is made by other user - Detects changes made by another user. See the example above.
- When changed to the same or different value - Detects any change.
- When changed to a different value only - Detects changes to a different value.
- Never - No conflict detection is applied to the property: the most recent update is written to the database.

Note: When implementing conflict detection in combination with web services, the so called entity-old part needs to contain the input for the conflict detector. If no values are passed via the entity-old part, the conflict detector skips the detection.

Available for use by others

Specifies whether the property will be available for use by other entities. See [Importing entities from other applications](#).

Guidelines for using this option with custom and subtyped entities follow:

- For a customized application, both a custom entity and its original entity must have the same setting. See [Customizing an application](#)
- For a subtype, this option can be enabled only if it is also enabled on its supertype. See [Advanced domain modeling using subtyping](#)

Type-specific attributes

You can also specify type-specific attributes by selecting a property in the list of building blocks. The available options depend on the type of the property. Some options become available after you select the property type, while others become enabled after you add the property and select it in the list of building blocks.

The following sections describe how to configure additional options for specific property types.

Boolean

When you create a Boolean property, you need to specify the name to display for the true, false, and none values. For example, you might want to display **Yes** for true and **No** for false.

You can also select an icon to represent each value.

To specify the display name and default value:

1. In the **Display Name** column, specify the name to display for each value.
2. In the **Default** column, select the value to be selected for the property when a new item is created. Selecting a default value is optional.

To delete the none value:

- Click the row for the **none** value and then click  (Delete).

To clear the default value:

- Click ... (More) > **Clear default value**.

To add a none value that you previously removed:

- Click ... (More) > **Add none value**.

To add an icon for a value:

1. Upload the image file to the project if it was not already uploaded. See [Specifying an image for a property](#).
2. Open the property in the list of building blocks.
3. Click the row that contains the value and select one of the following options:
 - Click ... (More) > **Browse to icon**.
 - Click  (Browse to icon) in the row.

To remove an icon for a value:

- In the **Property Details** pane, click the row that contains the value and select one of the following options:
 - Select the row, click ... (More), and then select **Clear icon**.
 - Select the row and click  (Clear icon).

Currency

- In **Currency**, select the currency. The list of currencies is taken from the ISO 4217 standard.
- In **Minimum value** and **Maximum value**, type the minimum and maximum values that a user can enter for the currency. These values must be within 19 digits of precision with 4 digits of scale.
- In **Default value**, type the value to supply for the value.

Decimal

- In **Max number of digits left of the decimal mark** and **Max number of digits right of the decimal mark**, type the maximum number of digits to the left and right of the decimal point. The sum of these values must be between 0 and 39.
- In **Min** and **Max**, type the minimum and maximum value that a user can enter.
- In **Default Value**, type the value to be supplied for the value.

Duration

- In **Default Value**, specify the default duration in Months, Days, Hours, and Minutes.

Float, Integer

- In **Min** and **Max**, type the minimum and maximum value that a user can enter.
- In **Default Value**, type the value to be supplied for the value.

Image

1. Upload the image file to the project if it was not already uploaded. See [Specifying an image for a property](#).
2. In **Default Image**, click **Browse**, select the image to associate with the property, and then click **OK**.
3. If you need to remove the selected image, click **Clear**.

Enumerated Text, Enumerated Integer

In **Enumeration Type**, select **Static** or **Dynamic**.

If you select Static:

- In **Value Length**, specify the maximum length of the values that will be entered in the list. (For Enumerated Text only.)
- In the **Display Name** list, enter a unique display name for the **none** value.
- If you want to add a new value, click **Add** and then specify the Display Name and Value. If you press ENTER, a new item is automatically created with the Value box selected. If you press ESC, no new items are created.

Select **Default** if you want the value to be selected by default when a new item is created. Selecting a default is optional.

To delete a static value:

- In the **Property Details** pane, click the row for the value that you want to remove and then click  (Delete).

If you delete an item and republish the entity after items have been created in Process Experience, the value that was deleted from the property is still displayed in the results panel. However, when you open an item that contains a deleted value, an error appears because the item contains a value that is no longer in the enumeration.

To change the display order of a static value:

- In the **Property Details** pane, click the row that contains the value and then click the Up or Down arrows.

To clear the default value:

- In the **Property Details** pane, click ... (More) and select **Clear default value**.

To add a none value that you previously removed:

- In the **Property Details** pane, click ... (More) and select **Add none value**.

To add an icon for an enumerated value:

- In the **Property Details** pane, click the row that contains the value and select one of the following options:
 - Click ... (More), and select **Browse to icon**.
 - Click the  icon in the row itself.

To remove an icon for an enumerated value:

- In the **Property Details** pane, click the row that contains the value and select one of the following options:
 - Click ... (More), and select **Clear icon**.
 - Click the  icon in the row itself.

If you select Dynamic:

Select one or both of the following options by clicking **Select** to select an existing business process or create a new one.

- In **Retrieve the values and display names for one item**, click Select, select the business process, and click OK. This business process is required.
- In **Retrieve the display name for a value when listing multiple items**, click Select, select the business process, and click **OK**. This business process is optional. If your application requires display names for the values, you must provide this business process.

To clear (deselect) the second business process:

- Click  (Clear).

See [Dynamic enumerations](#) for additional information about how to configure dynamic enumerations.

Text

- In **Default Value**, specify the default text.
- In **Pattern**, specify the text pattern to apply to the text. See [Text patterns](#).

Text patterns

It is useful to have application users enter some properties in a consistent way. A user can enter a telephone number, for example, in any of the following ways:

- 754-3010
- (541) 754-3010
- +1-541-754-3010
- 001-541-754-3010.

To constrain your application users to enter the telephone number as (541) 754-3010, you can add a pattern such as (999) 999-9999 to the Text property that contains the telephone number.

A pattern contains **9** to represent numeric characters and **Z** to represent alphabetic characters. Other numeric or alphabetic characters are refused. Non-numeric and non-alphabetic characters such as '(', ')', comma, space, and so forth are accepted. You can use characters "\", "[", "]", "{" and "}" but they must be prefixed with a backslash as they are reserved for future special purposes.

Enter a default value with the pattern applied.

In Process Experience, the telephone number is presented to users with the pattern applied. For internal use (for example, during modeling of a rule or in a web service operation), a value must be entered without the pattern applied. The previous telephone number would be used in a rule or web service operation as 5417543010

Note: When a text property with a pattern is used in an Email template, it is shown with the pattern applied. The pattern is not applied if an operation is performed with the property.

Example:

P1 is a text property with pattern '9999 ZZ'

P2 is a text property with pattern '(999) 999-9999'

The Email template body is entered in the entity modeler as follows:

Dear Sir / Madam,

We received the following account data:

Zip-code: {item.Properties.P1} (unformatted: {item.Properties.P1.substring(0)})

Telephone: {item.Properties.P2} (unformatted: {item.Properties.P2.substring(0)})

Kind regards,

ACME customer service

In Process Experience, with values for P1 and P2 added, the body is shown as follows:

Dear Sir / Madam,

We received the following account data:

Zip-code: 1234 AB (unformatted: 1234AB).

Telephone: (031) 040-54321 (unformatted: 03104054321).

Kind regards,

ACME customer service

Specifying an image for a property

When you create a [Boolean](#) or [enumerated](#) property, you can add an icon that represents the Boolean or enumerated value. A user can choose to show the icon in place of, or along with, a Boolean or enumerated value in a drop-list presentation of a form and a Column presentation option for list columns.

When you create an [image](#) property, you can select the image to associate with it.

To make an image available for selection, the image file must be uploaded to the project.

To add an image to a project:

1. Right-click the project and select **Upload document**.
2. Browse to the file, and click **Open**.
3. If necessary, change the name and description of the file and then click **Finish**.
6. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

7. If necessary, configure **Additional attributes** or **Type-specific attributes** for the property.
8. Click  (Save).

Dynamic enumerations

A dynamic enumeration enables the retrieval of enumeration values from an external source. The value list is retrieved in Process Experience through a business process. Inside the business process, you can invoke the web services needed to retrieve the values. The enumeration can be filtered based on the displayed entity's property values so the returned value list may vary based on the context.

A dynamic enumerated property enables you to select two business processes:

- The business process to retrieve the value list. This process is required.
- The business process to retrieve the display name. This process is optional. Display names are different from values so if your application requires display names you need to provide this business process.

See [Integrating BPM processes and entities](#).

The business process to retrieve the values and display names for one item

If a dynamic enumeration property is used in a form or list, the selected business process is invoked to present the enumeration values to the user. If the enumeration values depend on the context, you can pass the values of other properties of the same entity to the business process.

The namespace and the input and output messages for the business process must be defined as follows.

In these messages, `[SolutionName]` must be replaced by the name of the solution and `[EntityName]` must be replaced by the name of the entity. The solution name is constructed as described in [Entity web service details > Examples](#).

Namespace:

```
http://schemas/[SolutionName]/[EntityName]/enumeration
```

To define the namespace:

1. Right-click the BPM designer and select **Model Properties**.
The BPM properties open.
2. In the **General** tab, change the namespace to `http://schemas/[SolutionName]/[EntityName]/enumeration`.

Input message:

```
<GetEnumerationInputMessage xmlns = "http://schemas/[SolutionName]/[EntityName]/enumeration">
    <[EntityName] xmlns="http://schemas/[SolutionName]/[EntityName]">
        <[PropertyName]></[PropertyName]>
        <[PropertyName]></[PropertyName]>
    </[EntityName]>
</GetEnumerationInputMessage>
```

The `[PropertyName]` tag is the name of an entity property. These tags will receive the value of these properties that can be used by the business process to return a filtered enumerated list.

To define the input message:

1. In **Workspace Documents**, create a new XML Schema document in the same project, and provide the name as `[EntityName]`.
2. Copy and paste the following XML to the **Paste Schema or Instance** text area.

```
<[EntityName] xmlns="http://schemas/[SolutionName]/[EntityName]">
    <[PropertyName]></[PropertyName]>
    <[PropertyName]></[PropertyName]>
</[EntityName]>
```

3. Click **OK** and **Save**.
4. Go to the message map tab of the BPM designer.
 - In the source section, right-click **Process Specific messages** and select **Create Message**.
 - Provide the name of the message as `GetEnumerationInputMessage`.
5. In **Workspace Documents**, navigate to the created XML schema, and expand the **Elements** folder.
6. Drag the XML schema fragment that you created to the **GetEnumerationInputMessage** message in the BPM message map.

Output message:

```
<GetEnumerationOutputMessage xmlns="http://schemas/[SolutionName]/
```

```
[EntityName]/enumeration">
    <EnumerationValue>
        <Value></Value>
        <DisplayName></DisplayName><!--Optional field-->
        <IsDefault></IsDefault>
    </EnumerationValue>
</GetEnumerationOutputMessage>
```

The output message contains a collection of `EnumerationValue` elements that contain `Value`, `DisplayName`, and `IsDefault` tags. The `IsDefault` tag can be used to mark one value as the default value.

An example input message to the business process for the Order entity of the Order Management application follows:

```
<Order xmlns="http://schemas/MyCompanyOrderManagement/Order">
    <ShippingCountry>Japan</ShippingCountry>
</Order>
```

This might be used to retrieve the enumeration value list for the Vendor property based on the ShippingCountry property from the Order entity.

The business process to retrieve the display names for a list of values

The namespace and the input and output messages of the display name retriever business process must be defined as follows:

Namespace:

```
http://schemas/[SolutionName]/[EntityName]/enumeration
```

Input message:

```
<GetDisplayNameForEnumerationValueInputMessage xmlns="http://schemas/[SolutionName]/
[EntityName]/enumeration">
    <EnumerationValues>
        <EnumerationValue>S</EnumerationValue>
        <EnumerationValue>M</EnumerationValue>
        <EnumerationValue>L</EnumerationValue>
    </EnumerationValues>
</GetDisplayNameForEnumerationValueInputMessage>
```

Output message:

```
<GetDisplayNameForEnumerationValueOutputMessage xmlns=" http://schemas/
[SolutionName]/[EntityName]/enumeration">
    <EnumerationValue>
```

```

<Value>S</Value>
<DisplayName>Small</DisplayName>
</EnumerationValue>
<EnumerationValue>
  <Value>M</Value>
  <DisplayName>Medium</DisplayName>
</EnumerationValue>
<EnumerationValue>
  <Value>L</Value>
  <DisplayName>Large</DisplayName>
</EnumerationValue>
</GetDisplayNameForEnumerationValueOutputMessage>

```

In these messages, [SolutionName] must be replaced by the name of the solution and [EntityName] must be replaced by the name of the entity.

The EnumerationValues contain a collection of EnumerationValue tags in the input message that contain the Values of the dynamic enumeration property.

The output message has a collection of EnumerationValue tags, each EnumerationValue contains Value and DisplayName.

Notes:

- Properties of related entities cannot be used as context properties for the value list retriever business process.
- The ItemId is passed to the value list retriever business process as the rootEntityInstanceId of the instance properties. If the dynamic enumeration is used in a Create form, the ItemId will not have a value as the item is not yet created.
- The client locale is passed to the instantiationLocale property of the business process message map instance properties. To support localization, the business processes can use this instantiationLocale property to return the display names based on the locale value.
- When using a dynamic enumeration property in the Rule and List building blocks, you need to provide the value - not the display name.
- If the display name retriever business process is provided, the value list retriever business process should also contain the DisplayName in the output message.
- The value list retriever business process and display name retriever business process should always return the same display name for a given value. Otherwise, the user may see inconsistent display names in forms and lists.
- The performance of the Form, List filter, and Results panel is completely dependent on the execution time of the business process.
- If there is no display name retriever business process, the system cannot show display names for values after they are stored in the database. After storing a value, end users always see the actual value being stored and never a display name. If the value list retriever business process still returns display names, it would be inconsistent and confusing to show these display names at the time of selecting the value, and switch to the value later.

- If the enumeration values depend on the context, you can pass the values of other properties of the same entity to the business process.
- Dynamic enumeration properties that depend on the context (that is, the list of values depends on the values of other properties) cannot be used in the form of a related entity.
- To optimize performance, the response of the value list retriever business process and the display name retriever business process are cached for a user. This cache will be valid for 5 minutes. If modifications are made to one of the business processes, the new values will not be immediately visible. Workarounds are to log on as another user or restart TomEE.
- To improve performance of the business process, disabling the monitoring option is recommended. To do this, go to the business process Model Properties, select the Monitoring tab, and clear (uncheck) the following options:
 - Configure Monitoring on Process Level
 - Default Settings for all Activities

Limitation:

- In the List filter, the value list retriever business process will be invoked without any input properties even though the business process has an input message.

Null Values

Any property can have a null value if no value was assigned to it. Null values are handled gracefully by the expression engine and generally produce intuitive results. See [Expression language](#).

Validation

The system automatically performs a basic set of validations on all property types. This is the validation needed to ensure that the data entered by a user is correct for the property's data type. These validations are referred to as the property type's intrinsic validation. For example, the intrinsic validation of a numeric property ensures that a user cannot enter alphabetic characters, the intrinsic validation of a date property ensures that a user cannot enter an invalid date, and so forth. Intrinsic validation is not optional and is not configurable. Intrinsic validation honors the requirements of a user's locale.

Most applications need additional, solution specific, business logic validation. This type of validation often involves more than one property at a time. For example, an application may need to specify that the value of A + B cannot be greater than 10. This type of validation is handled by defining rules on the entity as a whole.

Identity

When you create an entity, an Identity building block is automatically added to it and shown in the list of building blocks. The Identity is stored in the database with a unique ID and is

used to identify the entity internally. You do not need to configure the Identity building block.

The Identity building block provides a property called ItemStatus. This property is used to implement draft items in a Create form in Process Experience. When a user creates a new item in Process Experience the ItemStatus property is set to 0 (Draft Item). When a user completes creating the item, the draft item is committed and the ItemStatus is changed to 1 to indicate that the item was created.

Use the itemStatus property as a condition in a rule if you want to prevent a rule from firing on a draft item.

Important: The ItemStatus property can impact rules that react to the **When a property changes** event, especially when creating rules for the Deadline building block. Since the Deadline timing does not start until the item is fully created, any rules created for the Deadline building block in a release that did not provide this property should be changed to include a condition so that the rules are not triggered if the item is in draft mode. The rules should test the Identity.ItemStatus property (ItemStatus is 0 for draft and 1 for created).

Notes:

- Draft mode is not supported for an entity that is external or imported from a database.
- Draft mode is not supported for an entity that is imported from a database and unlinked, if that entity does not have an auto incrementing primary key. For such an entity, the corresponding To Many repeating group container is displayed in Process Experience only when a Peer to Peer To Many relationship is placed on a Create form with a Repeating Group presentation.

Adding relationships

Entities in a business domain are often related to each other. For example, an Order Management application may consist of a variety of entities in addition to the main Order entity: order lines, customers, products, deliveries, and so forth. These entities do not exist in isolation but are related to the order and to each other in various ways. For example, each order needs to identify the vendor for the order, which products are included in the order, and how many of these products have already been delivered.

Types of relationships

You can add the following types of relationships to entities:

- Peer
- Parent-Child
- Reflexive

Peer

Peer relationships are used to relate separate, independent, entities to each other. Instances of both entities can stand alone, even if there are no instances of the related

entity. In an Order Management application, users must be able to just add a new customer to the system. There is no need to also immediately add orders for this customer. The orders will be added over time. The same is true for products - after adding new products to the system, it can take some time before customers start ordering them.

Parent-Child

Relationships in which an entity is owned or is a part of another entity are called parent-child relationships. Order line items can exist only within the context of an order. When deleting an order from the system, all of its order line items should also be deleted automatically as it makes no sense to keep the ordered items in the system.

Note: For a To Child relationship, the name of the child entity should be the same as the name of the relationship.

Reflexive

A reflexive relationship is a special case of a Peer relationship. The only difference is that there is only one entity involved and it is both the source and the destination of the relationship. One common example is an Employee entity that has a relationship to the employee's supervisor. A supervisor is also an employee who may have a supervisor. In this case, the Supervisor relationship is a To One relationship from Employee to Employee (it is a self-referencing, or reflexive, relationship). There may also be a paired To Many relationship called Reports that represents each supervisor's direct reports.

Relationship directions

Relationships can exist in different ways.

- In a One-Way relationship, only the source entity knows about the relationship and can access the details of the target entity. For example, if there is a One-Way relationship from the Order entity to the Customer entity, you can know which customer placed the order. With just a One-Way relationship there is no way to find all of a Customer's orders.
- In a Two-Way relationship, both entities are aware of the relationship and the relationship can be traversed in both directions. Making the Order-Customer relationship Two-Way would be required if you want to write rules to categorize customers based on the number and size of the orders they have placed in the last month.

Two One-Way relationships between two entities (in reverse directions) are not the same as a single Two-Way relationship. The key difference is that two One-Way relationships do not influence each other. That is, both relationships are populated and unpopulated independently of each other. With a Two-Way relationship, establishing a relationship in one way automatically also establishes a relationship in the reverse direction.

Multiplicity

Multiplicity refers to the number of instances of the target entity that can exist in the relationship. There are two types of multiplicities:

- To One - By setting the multiplicity of a relationship to To One, the system ensures that the relationship can contain at most one instance of the target entity. If another instance of the target entity is selected for that relationship, the system ensures that the previous instance is no longer related. For example, an Order entity might have a To One relationship to a Customer entity because each order can have only one customer. If another customer is selected for an order, the previous customer is no longer related.
- To Many - By setting the multiplicity of a relationship to To Many, the relationship can hold an arbitrary number of instances of the target entity. For example, adding a To One relationship to a form results in a different control compared to a To Many relationship. For example, a Customer entity might have a To Many relationship to an Order entity because each customer can have many orders.

Relationship possibilities

The following table describes the relationships that can exist between two entities:

From A to B	From B to A	Description
To One	No relationship	Each instance of A may refer to one instance of B, but there is no way to find out in an instance of B who is referring to it. Some instances of A may not reference an instance of B. Some instances of B may not be referenced by any instances of A. Often used for dynamic enumeration.
To One	To One (unpaired)	Each instance of A may refer to one instance of B, each instance of B may refer to one instance of B – but they do not have to be paired A1->B3->A5, etc. Some instances of A may not reference an instance of B (same for B to A). Some instances of A may not be referenced by any instances of B (same for B to A).
To One	To One (paired)	Each instance of A may be paired with one instance of B, and you can navigate back and forth from either side. Some instances of A may not reference an instance of B. Some instances of B may not be referenced by any instances of A.
To One	To Many (unpaired)	Each instance of A may refer to one instance of B. You can navigate from an instance of A to the instance of B that it refers to. You can navigate from an instance of B to a collection of instances of A but the A where you started navigating from could not be in this collection. Some instances

From A to B	From B to A	Description
		of A may not reference an instance of B. Some instances of B may not be referenced by any instances of A.
To One	To Many (paired)	<p>Each instance of A may refer to one instance of B. You can navigate from an instance of A to the instance of B that it refers to. You can navigate from an instance of B to a collection of instances of A. Some instances of A may not reference an instance of B. Some instances of B may not be referenced by any instances of A.</p> <p>Known limitation: If you have a To Many relationship between two entities, and want to pair it with a To One relationship, you cannot publish in between.</p> <p>Therefore, the following sequence will produce an error:</p> <ol style="list-style-type: none"> 1. Add a To Many relationship. 2. Publish. 3. Add a To One relationship and pair it with the To Many relationship. 4. Publish. <p>The following cycle will succeed:</p> <ol style="list-style-type: none"> 1. Add a To Many relationship. 2. Add a To One relationship and pair it with the To Many relationship. 3. Publish.
To Many	No relationship	Each instance of A may refer to multiple instances of B.
To Many	To Many (unpaired)	Each instance of A may refer to multiple instances of B. Each instance of B may refer to multiple instances of A. These are two independent "one to many" relationships.
To Many	To Many (paired)	Each instance of A may refer to multiple instances of B. Each instance of B may refer to multiple instances of A. This is a "many to many" relationship.
Child	No relationship	Each instance of A may own any number of instances of B. There will be no instances of B that are not owned by an instance of A.
Child	Parent	Same as above, but adds a To One relationship from B to its parent A. This relationship will be set automatically when B is created and cannot be changed.

Relationship names

The name of a relationship can be the same as the name of the target entity, although in many cases the names will not be the same because entities often play multiple roles in a business domain.

Consider, for example, the Employee entity. Some employees are the managers of other employees. In that case, one employee is playing the role of manager of other employees. The manager relationship is then likely to be paired with a relationship named "direct reports" in the reverse direction. If one employee is the manager of another employee, the latter employee is at the same time a direct report of the former employee. These two relationships go together. The only question is whether the business cares about the reverse relationship. If not, there is no business need to model it.

Also think about cases in which there are multiple relationships between the same two entities. For example, imagine that in an Order Management application you need to answer questions such as "How many orders did we recently receive from Europe?"

To answer such questions you can probably best introduce a separate Address entity and let every order have a relationship to Address. That would also make it very easy to distinguish between the address to which the order needs to be delivered (the delivery address) and the address to which the invoice needs to be sent (the invoice address). In this case, instances of the Address entity play different roles in the domain. In particular cases, that can even be the same instance.

Defining relationships

When configuring relationships, keep in mind that parent-child relationships are implicitly To Many. You cannot specify a To One multiplicity for a parent-child relationship.

To add a relationship:

1. In **Workspace Documents**, open the (source) entity for the relationship.
2. Click **Add > Relationship**.
3. In **Display Name**, type the name to be shown in Process Experience.
4. In **Name**, type a valid name.
A name can contain letters (including those from non-Latin alphabets), numbers, and underscore (_). The maximum length of the name is 128 characters.
5. Define the configuration options for the type of relationship you want to configure, as shown in the following table.

Type of relationship	Configuration options
One-Way Peer	<ol style="list-style-type: none">1. Select Type = To Peer.2. Select a Multiplicity:

Type of relationship	Configuration options
	<ul style="list-style-type: none"> ■ To One if instances of the source entity can be linked to at most one instance of the target entity. ■ To Many if instances of the source entity can be linked to multiple instances of the target entity. <p>3. Click the Browse button, select the target entity, and then click OK.</p>
Two-Way Peer	<ol style="list-style-type: none"> 1. Select Type = To Peer. 2. Select a Multiplicity: <ul style="list-style-type: none"> ■ To One if instances of the source entity can be linked to at most one instance of the target entity. ■ To Many if instances of the source entity can be linked to multiple instances of the target entity. 3. Click the Browse button, select the target entity, and then click OK. 4. In Pair with, select an existing relationship, if present, that is defined from the target entity to the current entity.
One-Way to Child	<ol style="list-style-type: none"> 1. Select Type = To Child. The Multiplicity is automatically set to To Many. 2. Provide a Child name and a Child display name. All child entities of the same parent must have unique Child names.
Two-Way to Parent	<ol style="list-style-type: none"> 1. Select Type = To Parent. This option is available only for child entities in one-way relationships where Type = To Child. <ul style="list-style-type: none"> ■ The Multiplicity is automatically set to To One. ■ The Target Entity is automatically set to the name of the parent entity. ■ Pair with is automatically filled in with the name of the one-way relationship between the parent and the child. <p>Note: This type of relationship can be added only once.</p>
Reflexive	<ol style="list-style-type: none"> 1. Select Type = To Peer. 2. Choose the multiplicity you want. 3. Select a Target Entity by clicking Browse. 4. Open the same entity as the source entity and then click OK. 5. In Pair with, select an existing relationship, if present, that is defined from the target entity to the current entity.

6. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

7. Click  (Save).

To make a relationship available for use by other applications:

1. Select the relationship in the list of building blocks.
2. Select **Available for use by others**.
3. Click  (Save).

To specify the delete behavior of the related item(s):

1. Select the relationship in the list of building blocks.
2. In **Delete behavior**, select **Yes** or **No**.
3. Click  (Save).

Deleting relationships

If a user deletes a related item or clears a relationship in Process Experience (for example, in an item, list, or form), the associated data is handled based on the type of relationship and whether the Clear or Delete option is used. For example, assume that a form contains a property called Color from a related entity.

- The **Clear** option is available for To One/To Peer and To Many/To Peer relationships. If a user clicks **Clear**, the relationship is broken but the related value is not deleted. If a user clears the Red value, that value is no longer associated with the item but it is still available for selection for that item and other items of that type.
- The **Delete** option is available for To Many/Parent Child relationships. If a user clicks **Delete**, both the relationship and the related value are deleted. For example, if the user deletes the Red value, that value is no longer available for associating with existing items or for creating new items.

Note: When the **Delete** option is set to yes, a dangling reference remains when the related entity of the unpaired (ToOne/ToMany) relationship is deleted. Also, the relationShipChanged rule is not triggered.

If you delete a relationship from an entity in CWS, it is deleted from any existing items that are associated with it and not available for creating new items. For example, it is removed from any lists and forms that included it.

Tracking history

Add the History building block to an entity to enable tracking activity in instances of the entity to record an audit trail. An item's history is a collection of history events. Each event

records a specific change or access to items.

When an entity includes a History building block, an option called History is available for adding to an action bar. See [Creating an action bar](#).

To add history to an entity:

1. In **Workspace Documents**, open the entity.
2. Click **Add > History**.

Important: The history log must be defined in the same project where it is used with the history building block.

3. In the **History Log** area, click **Browse** and select the history log where this entity's history will be recorded or create a new history log as follows:
 - a. Click **New > History Log** at the bottom of the dialog box.
 - b. In the **History Log Properties** panel, type a **Display Name** and **Name** for the history log.
 - c. In **Compression Type**, select **None**, **Deflate**, or **GZip**.
 - d. In **Description**, type a description of the history log.
 - e. Click  (Save).
 - f. If necessary, make any changes in the summary box and then click **OK**.
4. If you want every user access to an item to be recorded, select **Record access events**.
5. In the **On Deletion** list, specify what happens to an item's history when it is deleted.
 - Discard history - The item's history is discarded completely.
 - Retain history - The item's history is retained in its entirety.
 - Only note deletion - The item's history is discarded and replaced with a single history event noting that the item was deleted.
6. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

7. Click  (Save).

To rename a history log:

1. Open the History building block.
2. In **History Log**, click **Browse**.
3. Click the down arrow next to the name of the history log and select **Rename**.
4. Type the name and click **OK**.

To delete a history log:

1. Open the History building block.
2. In **History Log**, click **Browse**.
3. Click the down arrow next to the name of the history log and select **Delete**.
4. Type the name and click **OK**.
5. Click **Yes** when prompted for confirmation.
A message appears if the history log is in use by other entities.
 - Click **Yes** to delete the history log.
 - Click **Show Used by** to see where the history log is used.

Configuring security

To provide different levels of access to different groups of users, you can use the Security building block.

When a solution is deployed, it is closed to all users by default. A solution is first protected by the Use permission on the solution as a whole. Once a user has been granted Use permission to the solution, that user has full access to any unprotected entities (entities that do not have a Security building block) in that solution. This enables rapid creation of simple solutions.

In most organizations, not all users should be able to see all data or use all functionality. Access should be limited and only given based on the user's role. Role based security provides a way to give access to entities or building blocks in a more detailed level. For example, you may need to specify that only supervisors should be allowed to see employee records or that only managers should be allowed to see and approve purchase orders over a certain currency value.

Important: When a Security building block is added to an entity, Process Suite users can perform only those operations on that entity that are granted to them in the Security Editor.

Permission assignments are horizontal and should be aligned across the entire application. It does not make sense for a user to have Create permission for forms but no permission for lists.

Entity permissions

The following table describes the permissions that can be configured on entities using the Security Editor.

Entity Permission	Description
Create	Users can create new instances of the entity.

Entity Permission	Description
View	Users can see that a specific instance exists. This permission does not allow users to see the properties and relationships of the instance. To do that, specific read permissions on the properties and relationships are needed. This view permission allows the instance to be included in, for example, a list result. It is similar to a gateway-permission on the entity. Without it, the entity cannot be used for anything, despite any other permissions granted. In previous releases this permission was called the read permission.
Delete	Users can delete the complete entity with its properties. To be able to delete, the user also needs View permission.

Building block permissions

The following table describes the permissions that can be configured on building blocks using the Security Editor.

Building block	Permission	Description
Assignee	Execute	Designates the user, role, or organizational unit that is responsible for the entity. See Assignee task permissions .
Business Workspace		Enables entities with extended ECM capabilities and helps users to manage workspace and content in the applications. See Business workspace task permissions .
Content		Enables users to add multiple documents to an entity and manage documents through various actions exposed by the building block. See Content task permissions .
Deadline	Update	Enables setting or updating a deadline.
Discussion	Execute	Enables actions for adding a comment. See Discussion task permission .
Email	Send Email	Users can send email.
Email template	Read	Users can use a specific email template to compose an email.
File		See File task permissions .
Lifecycle	Execute	The Lifecycle building block adds a case model to an entity, making the entity into a case. The Lifecycle building block enables specifying permissions for

Building block	Permission	Description
		individual tasks. Lifecycle also exposes actions (user events) that can be set with execute permission. See Lifecycle task permissions .
List	Use	Enables the list to be used in Process Experience.
Property	Read	Users can read the value of the specific property of the entity. If a Read permission is blocked on the property, a property level run time only security group permission will be deployed to block read access.
	Update	Users can update the value of the specific property of the entity. If an Update permission is blocked, a property level run time only security group permission will be deployed to block update access. To be able to update, the user also needs Read permission.
Relationship	Read	Users can retrieve the identifier of the target entity instance via the source entity instance. Without Read permission on the relationship, the related target entity instance cannot be accessed from the source entity instance. This permission does not allow users to read properties or relationships of the target entity instance. To do that, specific read permissions on these properties and relationships are needed.
	Update	Users can make the source entity instance relate to another target entity instance. To be able to update a relationship, the user also needs Read permission on that relationship. For To Many/Parent Child relationships, the permissions are set on the Child entity, not on the relationship. This Update permission does not allow users to update properties or relationships of the target entity instance. To do that, specific update permissions on these properties and relationships are needed. Note: For paired relationships, keep in mind that the security for both relationships should be set equal to have no security 'holes'. Example: Suppose entity B has a relationship (to many to entity A) without update permission and entity A has a paired relationship (to many to entity B) with update permission. This means

Building block	Permission	Description
		that you cannot add instances of A to an instance of B, but you can add instances of B to an instance of A: So effectively you can still add an instance of A to an instance of B.
Rule	Update	This permission applies only to rules of type When a user triggers an action button . It enables the user to trigger a rule action. The action button is only available when the user has permission and when the expression in the rule evaluates to true.
Title	Execute	Enables user to update the title description of the entity.
Tracking	Update	Adds information to an entity to track when and by whom an instance was created and last modified. Each entity can include only a single Tracking building block. Tracking adds two relationships (CreatedBy and LastModifiedBy) for which permissions need to set.
Web service	Use	Enables the user to use a Find web service operations. Note: The Web service building block exposes permissions only for the Find web service operations. All users can use the other web service operations, but the authorization check is done at the entity level. The entity, property, and relationship permissions are applied to the result.

Note: It is possible for a Find web service operation to use an expression with a property for which you do not have Read permission. Therefore, a Find web service operation has an explicit Use permission that overrides the Read permission that the user has on the specific properties used in the expression. The Read permissions on the properties are, however, applied to the query result and are not shown in the Find web service operation response.

Example

Assume that an entity called Employee has properties called Name and Salary. A Find web service operation is defined as follows:

Find all Employees with Salary > 100000.

If you have no Read permission on the salary but you do have Use permission on the Find web service operation, the Find web service operation returns all names of the employees having a salary > 100000. However, the employee salaries are not shown in the response.

Assignee task permissions

The following table describes the permissions that can be configured on assignee tasks using the Security Editor. To work with assignee actions, you must set permissions for the AssignIdentity and GroupIdentity manually.

Action/permission	Description
Assign	Users can assign responsibility for an item that is currently unassigned. If the assignee is set to a work group (Role or Organization unit), the Assign action enables a user (with Assign permission) to assign an item to a user from that workgroup.
Forward	Users can assign responsibility for an item that is currently assigned to either the user individually or to one of the user's workgroups or to another user or workgroup. This action displays a form prompting for the workgroup or individual to be set as the assignee.
Revoke	Users can revoke a claimed entity item. Once revoked, the item moves back to the corresponding workgroup. Only claimed items can be revoked.
Claim	Users can accept responsibility for an item that is currently unassigned or assigned to one of the participant's workgroups, setting the assignee to the participant specifically.
AssignIdentity	Updates the relationship to hold identity for users.
GroupIdentity	Updates the relationship to hold workgroup information.

Business workspace task permissions

The following table describes the permissions that can be configured on business workspace building block tasks using the Security Editor.

Action/permission	Description
Synchronize Workspace	Enables creating and updating of Business Workspace in Content Server with the configured entity properties and relationships.
Open Workspace	Users can open the Extended ECM user interface to access, navigate, and perform all document management and workspace operations in the business workspace in the same way as a Content Server user performs these operations.
Add business attachment	Users can attach an existing document in Content Server to the entity instance and workspace.
Delete business	Users can delete an attachment added to the entity and

Action/permission	Description
attachment	workspace.
Open in Brava Viewer	Users can view attachments in the Brava viewer. This task is available only when the document store connector is configured with Brava.

Content task permissions

The following table describes the permissions that can be configured on Content building block tasks using the Security Editor. For the Content building block, the permissions can be configured at folder level. The list of tasks depends on the repository or document management system configured with the document store connector. For example, if the document store connector in Process Platform is configured for Content Server, users will see check in, check out, and other actions that are applicable to Content Server.

Action/permission	Description
View folder	Users can see the folder and its contents.
Upload	Users can upload a file.
Download	Users can download a file attached to the entity.
Check in	Users can check in a version of the file.
Check out	Users can check out a file.
View Versions	Users can see all the versions of a file.
Audit	Users can see the audit history of actions.
Search	Users can search for files in the configured repository.
View properties	Users can see metadata properties of a document.
Update properties	Users can update metadata properties of a document.
Copy attachment	Users can copy an existing document from repository and attach it with entity instance.
Link attachment	Users can link or refer an existing document from repository with entity instance.
Move	Users can move document from one folder to another.
Share in repository	Users can share document with one or more users of the repository.
Public URL	Users can generate a public link for the document.
Delete versions	Users can delete a version of the document.
Open in Brava viewer	Users can open the document in Brava viewer. This task is

Action/permission	Description
	available only when the document store connector is configured with Brava.

Discussion task permission

The following table describes the permission that can be configured on the Discussion Add Comment task using the Security Editor.

Action/permission	Description
Add Comment	Users can add comments to a discussion.

File task permissions

The following table describes the permissions that can be configured on file tasks using the Security Editor. The list of tasks depends on the repository or document management system configured with the document store connector. For example, if the document store connector in Process Platform is configured for Content Server, users will see check in, check out, and other actions that are applicable to Content Server.

Action/permission	Description
Upload	Users can upload a file.
Download	Users can download a file attached to the entity.
Check-in	If Content Server is used, users can check in a version of the file.
Check-out	If Content Server is used, users can check out a file.
View versions	Users can see all the versions of a file.
Audit	Users can see the audit history of actions.
Search	Users can search for files in the configured repository.
View properties	Users can see metadata properties of a document.
Update properties	Users can update metadata properties of a document.
Copy attachment	Users can copy an existing document from the repository and attach it to an item.
Link attachment	Users can link or reference an existing document from the repository with an item.
Share in repository	Users can share a document with one or more users of the repository.
Public URL	Users can generate a public link for the document.

Action/permission	Description
Delete versions	Users can delete a version of the document.
Open in Brava viewer	Users can open the document in the Brava viewer. This task is available only when the document store connector is configured with Brava.

Lifecycle task permissions

The following table describes the permissions that can be configured on lifecycle tasks using the Security Editor.

Action/permission	Description
Lifecycle - Static action Add Task	Users can plan tasks for an entity item.
Lifecycle - Dynamic actions User defined actions	User events defined in the Lifecycle model will be presented as actions on an entity instance and these actions can be performed by user. These actions will be available based on the state of a Lifecycle. Action names displayed are in this format: action description (action name).
Lifecycle task - Static actions	
The following actions will be available based on the state of a task	
Add related task	Users can plan intermediate activities for a selected task.
Assign	Users can assign a task to a user. Once a task is assigned to a user, the task becomes a personal task for the user.
Claim	Users can claim the task and work on it.
Complete	Users can complete a task to finish executing the task.
Delegate	Users can delegate a task to other users.
Forward	Users can forward a task from one user to another or across teams, work lists, and roles.
Modify due date	Users can modify due date of a task.
Modify start date	Users can modify the start date of a task.
Pause	Users can temporarily stop executing the task.
Resume	Users can resume a paused task.
Revoke	Users can revoke a claimed task.
Skip	Users can skip a task. You can skip tasks before they are

Action/permission	Description
	completed if required. Once the task is skipped, it becomes a non-workable task.
Start	Users can start executing the task.
Stop	Users can stop executing the task.
Suspend	Users can suspend a task from executing. Only a work list manager or team lead can suspend the execution of a task in the work list or team folder.

Setting permissions

To set permissions, you need to add the Security building block to the entity. This enables you to define security on entities, their properties, relationships, actions, and other rights available on the various building blocks.

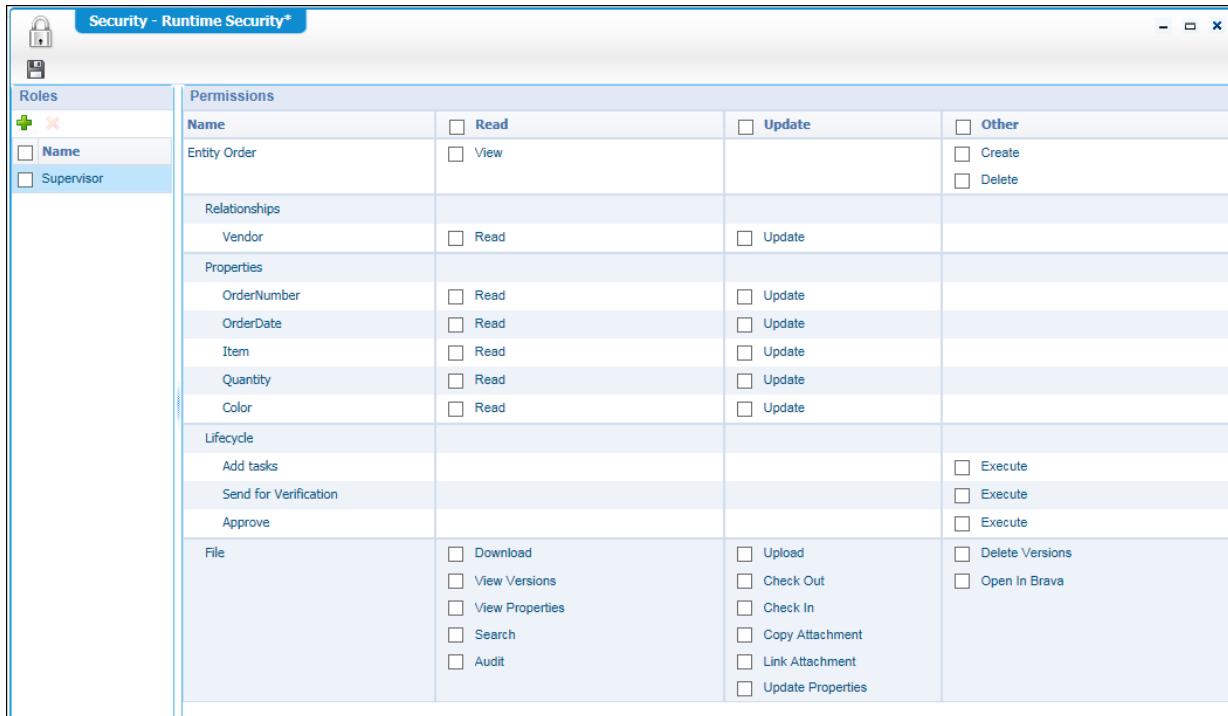
Before you begin, be sure that the necessary application roles are created in your project.

To add security for an entity:

1. Open the **Workspace Documents** folder and be sure that it contains at least one entity.
2. Open the entity.
3. Click **Add > Security**.
6. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

7. Click **Configure**.
8. In the Security Editor, grant permissions to roles. The permissions are grouped for the entity and per building block added to the entity.
9. Click  (Save).



The screenshot shows the 'Security - Runtime Security' editor window. On the left, there's a 'Roles' panel with a '+' icon for adding new roles and two existing roles: 'Name' and 'Supervisor'. The main area is titled 'Permissions' and contains a table with several sections:

	<input type="checkbox"/> Read	<input type="checkbox"/> Update	<input type="checkbox"/> Other
Name	<input type="checkbox"/> View		<input type="checkbox"/> Create <input type="checkbox"/> Delete
Entity Order			
Relationships			
Vendor	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
Properties			
OrderNumber	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
OrderDate	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
Item	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
Quantity	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
Color	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
Lifecycle			
Add tasks			<input type="checkbox"/> Execute
Send for Verification			<input type="checkbox"/> Execute
Approve			<input type="checkbox"/> Execute
File	<input type="checkbox"/> Download <input type="checkbox"/> View Versions <input type="checkbox"/> View Properties <input type="checkbox"/> Search <input type="checkbox"/> Audit	<input type="checkbox"/> Upload <input type="checkbox"/> Check Out <input type="checkbox"/> Check In <input type="checkbox"/> Copy Attachment <input type="checkbox"/> Link Attachment <input type="checkbox"/> Update Properties	<input type="checkbox"/> Delete Versions <input type="checkbox"/> Open in Brava

To create a role:

1. In the **Roles** panel, click  (Add).
The **Select Role** dialog box opens.
2. Click **New > Role**.
3. Type a name and description of the role.
4. In **Type**, select **Functional**, **Application**, or **Internal**. Each type of role is handled in the same way from a security perspective.
 - Functional roles are high-level roles that can be created based on the needs of the organization.
 - Application roles provide access to a set of resources within one package.
 - An internal role has specific access privileges that are required for an activity.
5. Click  (Save).
6. When prompted, confirm the information you entered and click **OK**.

To grant permissions to a role in the Security Editor:

1. In the **Roles** panel, select a role or click  (Add) to create a new role.
2. In the **Permissions** panel, select each permission you want to grant to the selected role.
Using the check boxes in the table header row, you can grant or deny all permissions within that column with a single click. The appearance of these check boxes also indicates whether all, none, or at least one permission in that column has been granted.

Note: Some permissions are implied. For example, if you select Delete permission for an entity, View permission is selected automatically. For properties and relationships, if you select Update permission, Read permission is selected automatically.

3. Click  (Save).
4. Close the Security Editor.

To update permissions to a role in the Security Editor:

1. In the **Roles** panel, select the role.
2. In the **Permissions** panel, select or clear the permissions you want change.
3. Click  (Save).
4. Close the Security Editor.

To delete the security for a role in the Security Editor:

1. In the **Roles** panel, select the role.
2. Click **Delete role**.
3. Click  (Save).
4. Close the Security Editor.

To refresh the defined permissions in the Security Editor:

- Click  (Refresh) to refresh the list of building blocks and modified permissions.
- If you have unsaved changes, you are prompted to save them.
 - If you click **Yes**, your changes are discarded.
 - If you click **No**, your changes are not discarded.

To remove security from an entity:

1. In **Workspace Documents**, open the entity.
2. Position the pointer over **Security**.
3. Click **Delete**.
4. Click  (Save).

Conditional permissions

You can grant permissions based on a specific condition. The permission is granted only when the condition is satisfied. In the following example, the user or role with Update permission is allowed to update the property when the value of Amount is less than 100.

grant: Update, condition: Amount is < 100

Conditions have a name and are defined as an expression. You can re-use them to grant multiple permissions.

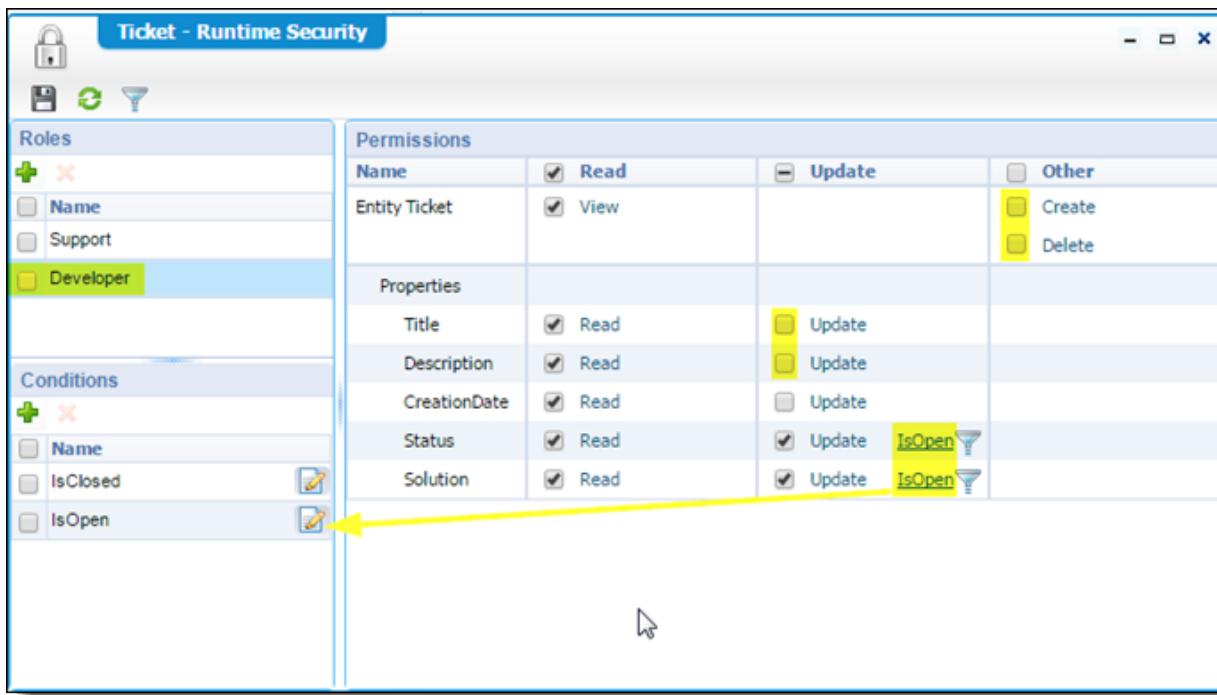
Example

Assume that your organization uses a ticket system to report bugs. An entity named Ticket was defined that has roles called Support and Developer. It also has two conditions called isOpen and isClosed referring to the Status property. The isOpen condition is true when Status equals Open and the isClosed condition is true when Status equals Closed.

Support can delete a ticket only when the Status is equal to Closed and the Solution can only be updated as long the Status is equal to Open. A developer has limited permissions and is allowed to change the Status and Solution only when the ticket Status is Open.

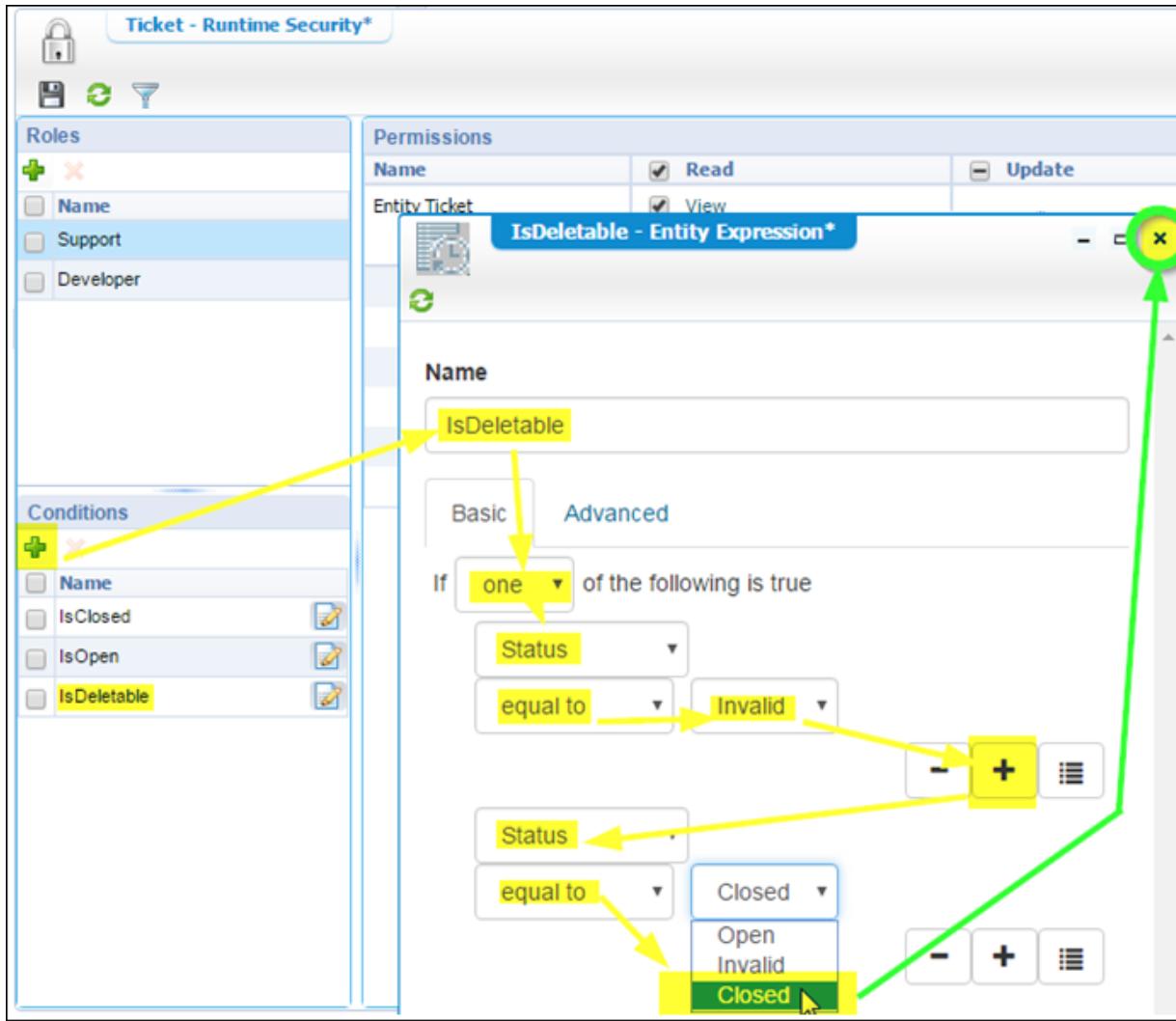
Name	Read	View	Create	Update	Other
Entity Ticket	<input checked="" type="checkbox"/>				
Properties					
Title	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Description	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	
CreationDate	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	
Status	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	
Solution	<input checked="" type="checkbox"/>				

Condition 'isOpen': item.Properties.Status=="Open"



Creating conditions

You can create multiple conditions. The conditions are stored in the context of a single entity. They can be reused on all configured permissions for that specific entity and all the derived types for that specific entity (subtypes and customized types). Within the conditional expression, all the specific properties defined on that entity can be used. At runtime, when an expression is evaluated, the property values of the current instance of the entity are used. A generic (re-usable) entity expression editor is used to define the expression. The same editor is used to define rules. For example another status named Invalid can be added and the Support engineer is allowed to delete tickets with the status Closed or Invalid. In that case, you could create a new condition named IsDeletable.

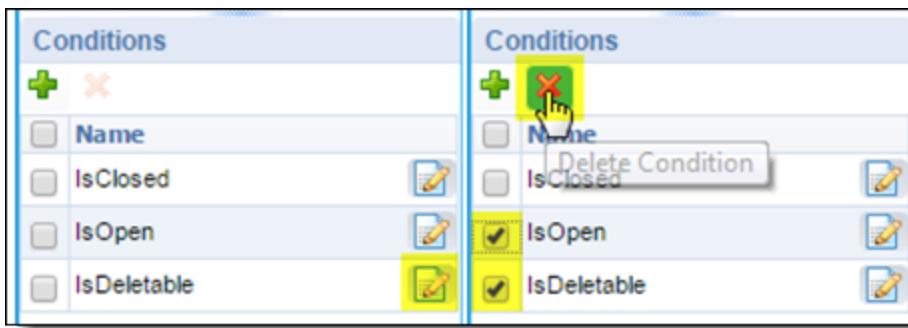


Notice that you need to click the (x) button in the upper right corner of the dialog box to finish the definition of the expression. The expression is saved as soon you click **Save** in the ACL Editor or in the Entity Modeler.

Modifying or deleting a condition

To see the details of a condition or to modify a condition, click (Edit) next to the condition name. This opens the expression editor with the condition details.

To delete a condition, select the check box that precedes its name and click (Delete Condition).

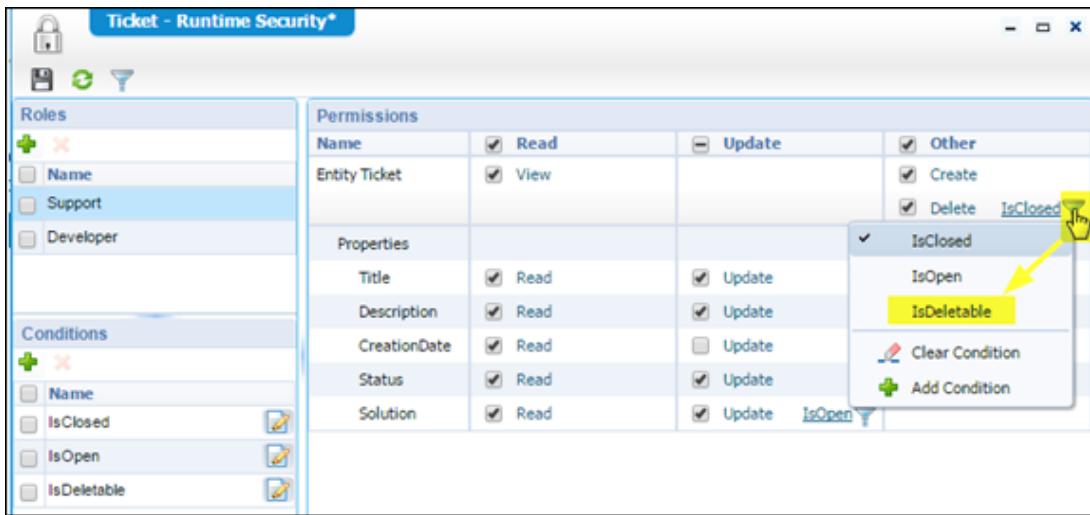


Assigning a condition to a permission

Conditions can be used for permissions that are granted. Once a permission is granted, a filter icon () is displayed next to the permission. The icon is displayed only when the pointing device is positioned over the permission cell. Not all rights allow you to assign a condition. You can assign a condition to a permission only when the underlying model supports it.

When you click the filter icon (), a context menu appears where you can select one of the defined conditions. The current assigned condition is shown with a check mark. In the same context menu you can also unassign the condition using the Clear Condition () option.

The condition is only removed from the permission, but the condition itself remains in the list of all available conditions for later use. You can also create a new condition from the same context menu (Add Condition).

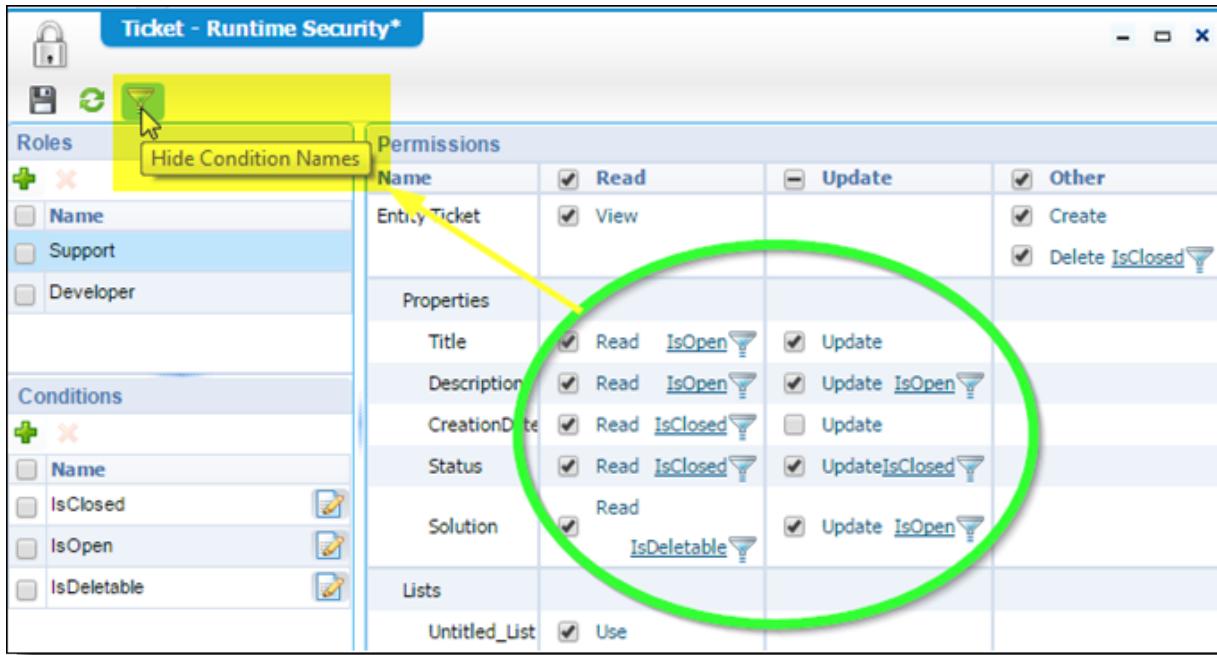


You cannot assign a condition to the following permissions:

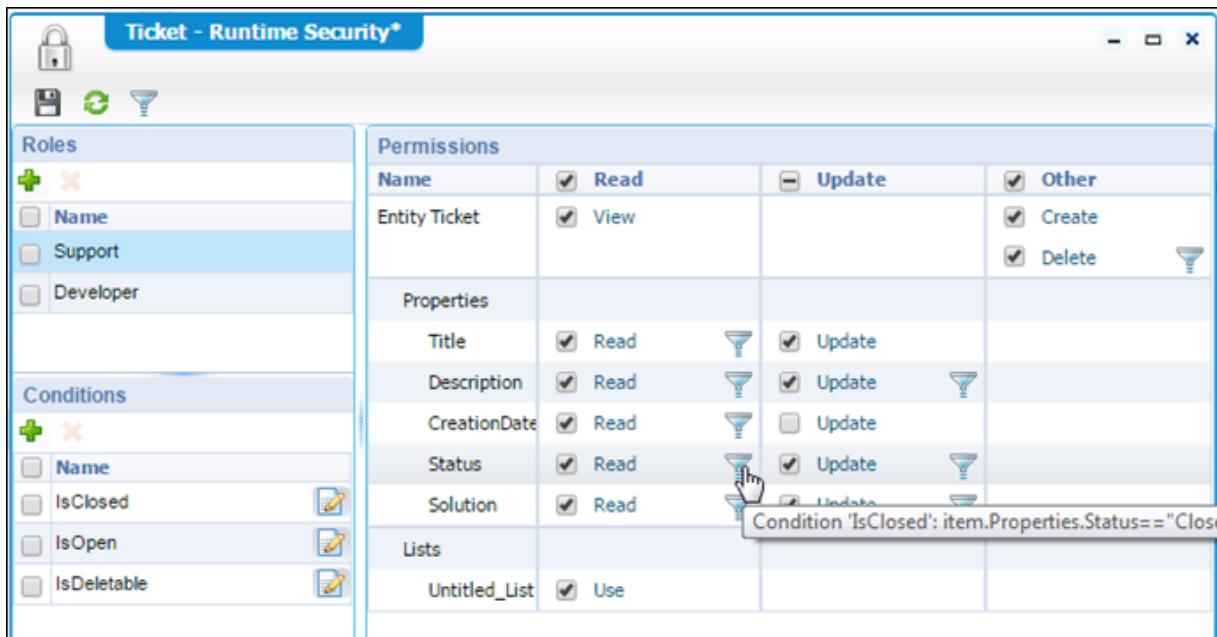
- Entity – Create
- List – Use

Hiding condition names from permissions

When your Permissions pane becomes crowded with (long) condition names, you might lose the overview. In that case, you can hide the condition names with the filter toggle button in the toolbar.



This option gives a simplified overview of the rights and only the filter icon remains to indicate that a condition is assigned to a right. The tooltip for the condition is still available.



Customization and subtyping

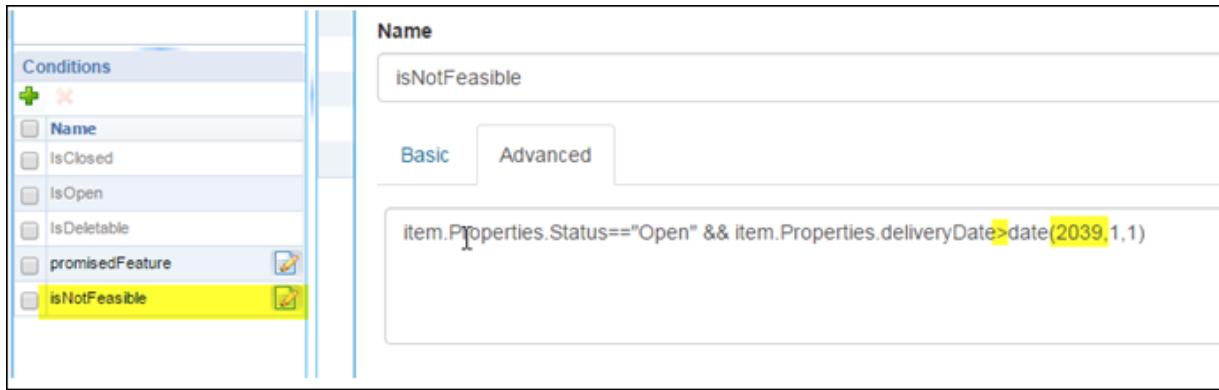
Assume that you create a subtype called BacklogTicket with an additional property called deliveryDate. That subtype entity inherits all the conditions defined on the supertype. The conditions from the supertype are indicated with light gray highlighting. The inherited conditions can be used, but cannot be modified. It is however possible to create new conditions where you can re-use all the inherited properties. In the context menu, you can assign the newly defined conditions on the subtype, but you can also re-use all the inherited conditions. When you customize a permission that was given in the supertype, either via the check box or condition, you see the revert icon (). You can always revert and go back to the permission as it was defined on the supertype (with or without condition).

Name	Read	Update	Other
Entity BacklogTicket	<input checked="" type="checkbox"/> View		<input checked="" type="checkbox"/> Create <input checked="" type="checkbox"/> Delete
Properties			
Title	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	
Description	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	
CreationDate	<input checked="" type="checkbox"/> Read	<input type="checkbox"/> Update	
Status	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	
Solution	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	
deliveryDate	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	

Complex conditions

The condition expressions also can be more complex. For this, you can also use the Advanced view in the expression editor.

Here is another example: the “isNotFeasible” condition



Security for ad hoc viewers

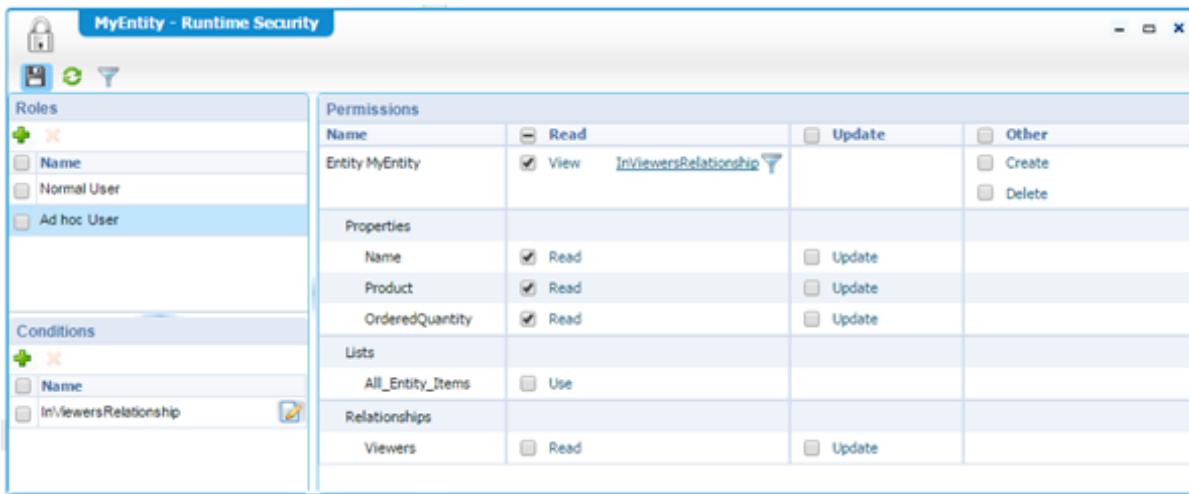
Use the Security building block to configure read-only permission for viewing entity data. To do this, you can configure permissions for a certain role in such a way that only read access to the applicable properties is granted. Any user who has that role assigned is able to view the entity data.

To configure security for ad hoc viewers:

1. Define a role for the users who are eligible to receive a link to view the entity (for example, ad hoc user). This role should have the Entity Runtime User as a sub role using CWS a runtime reference.
2. Add a To Many relationship from your entity to User (from the Identity package) and name it, for example, Viewers.
3. Define conditional security.
 - For the Ad hoc user role, grant Read permission for any applicable property and other building block.
 - Add the following condition (for example, InViewersRelationship) to the View permission for the entity (where Viewers is the name of the relationship):

(item.Viewers [].Properties.UserId) .contains (USER.Properties.UserId) :

The security configuration is similar to the following example:



- Assign the users (who should be able to view only) to the (for example) Ad hoc user role, and add the user-to-view to the Viewers relationship just before you distribute the link. In this way, Ad hoc users will have read-only access to items for which they have been explicitly defined as Viewer.

How conditions are applied in item creation

When creating an item, the properties having a conditional permission are not shown on the form if the condition of the permission has not been met. To have all properties visible during creation of an item, you need to extend the condition with: `or item.Identity.ItemStatus==0` (indicating that the item has a status of Draft).

If a role has a condition assigned to the Entity – View permission and that role also has permission granted for Entity – Create, you need to extend that condition with `or item.Identity.ItemStatus = 0` (indicating that the item has a status of Draft). This avoids an error being raised in Process Experience when a user creates a new item.

How conditions are applied in list results

For any value of a property to be displayed in a list result, the Read permission for the property must be granted. If the Read permission has a condition, the expression must be evaluated to grant the permission. Any property that is included in a list is available for evaluation by an expression, even if the property is hidden. If a property referenced in an expression is not included in a list, the property is not available for the expression to evaluate it correctly. As a result, the expression cannot be evaluated and the user cannot see the values of this property for any row in the list.

For conditions of Read permissions on properties that are in the entity on which the list is defined, the referenced properties in that entity are automatically included in list, so you don't have to explicitly add these to your list. For any conditions on permissions for any other entities (subtypes or relationships), the referenced properties are not automatically included in the list, and you may need to explicitly add them. Otherwise, the expression cannot be evaluated and the user cannot see the values of the property for any row in the list.

If a list includes values for a property in a related entity, all of the following security permissions must be granted:

- Read Permission for each relationship in the relationship path.
- View permission for each target entity of the relationship in the relationship path.
- Read permission for the property.

The following scenarios provide some examples of how conditions are applied in lists:

- Entity A has a security policy with conditional permission on a property. If the list does not include the property referenced in that condition, the system automatically detects that the property is included in the results because it is included in the entity on which the list is defined.
- A list is configured on Entity A, but its subtype A2 replaces the security policy and adds a conditional permission on the Read permission of the property with an expression that references a property in Entity A (the base entity). If the property referenced in that expression is not included in the list, the property is not available for the expression to evaluate it correctly. As a result, the expression cannot be evaluated and the user cannot see the values of this property for any row in the list. When you explicitly add that property to the list, the expression can be evaluated.
- In the previous example, if the new Read permission in subtype A2 references a property defined in A2 (the subtype entity), the property is not visible in Entity A and cannot be added to the list. As a result, the expression cannot be evaluated and the user cannot see the values of this property for any row in the list of type A2.
- Entity A has a To One relationship to Entity B. In Entity B, the security policy has Read permission on properties that have conditions and those conditions reference properties in Entity B. If a property referenced in the expression is not included in the list, that property is not available for the expression to evaluate correctly. As a result, the expression cannot be evaluated and the user cannot see the values of this property for any row in the list. When you explicitly add each referenced property to the list, the expression can be evaluated.
- Entity A has To One relationship to Entity B. B has subtype B2 that replaces the security policy from Entity B and adds a condition for Read permission on a newly-added property in subtype B2. If the property referenced in that expression is not included in the list, the property is not available for the expression to evaluate correctly. As a result, the expression cannot be evaluated and the user cannot see the values of this property for any row in the list of type B2. Since you cannot explicitly add the property to the list because it is not visible in Entity B when you are defining the list, there is no workaround.

By default, a list retrieves 200 items per page. If a condition on Entity - View permission has been assigned, the specified number of items matching the condition are shown on each page. This can be fewer than 200 items. A user needs to go to the next page to see the next batch of items matching the condition. Sometimes no items will be shown on a page, but items matching the condition will be shown on a next page. To avoid this behavior, define a filter for the list that is similar to the expression of the condition.

Chapter 4

Defining the process flow

Process flow building blocks define how information flows through a business process.

Configuring a lifecycle

Using the Lifecycle building block, you can define how an entity instance evolves as it moves through business operations. The core components of a Lifecycle model are states, transitions, events, and activities. Each state represents a possible stage in a Lifecycle and each state transition can have associated activities that are started when the state transition occurs. The transition occurs when the transition events are initiated. Transition events can refer to conditions that are evaluated when properties of the entity are changed, when actions are performed by users, and when activities are completed by users or the system.

Important: If any of the entities in your application include the Lifecycle building block, you must activate the InBox Task list so that the list of user tasks will be shown in Process Experience. See [Defining security for Inbox Task Management](#).

Caution: In entity forms, the LifecycleTask relationship from Lifecycle is read only. Performing actions such as Create and Delete on the LifecycleTask Repeating Group Container will lead to unexpected errors.

Using the Lifecycle building block, you can build solutions that are both case-centric and process-centric.

Case-centric solutions

Case-centric solutions are case and knowledge driven, unstructured and collaborative, and suited to knowledge workers.

Examples

- Distinguishing between routine and non-routine cases, for example for credit assessment or claims management.
- Doing something extra for the customer, such as offering an insurance product during claims management.
- Resolving disputes. For example, processing payments is a conventional structured process, but when a customer disputes a charge or demands a refund, case management is usually required.

Process-centric solutions

Process-centric solutions are task driven, structured and standardized, and suited to back office operations.

Example

- Manufacturing products in accordance with orders that are received.

States

States model the functional stages of an entity by grouping activities that can be executed while the entity instance is in that specific state. States are normally action words such as planned, approved, or submitted. States can be nested. That is, an item can be in a child state and at the same time in the parent state. In case of multi-level nested states, the Lifecycle instance is always one of the innermost state. A Lifecycle model always has an initial state and usually one final state. The Lifecycle model itself is the top-level state, so activities can be part of the Lifecycle model but not part of any of the modeled states.

You can define various states in the Lifecycle model. The state is available as a system property that can be used to determine the exact status of the entity instance at any point in time.

When defining rules, you can use the status properties as part of the expressions used for defining the rule condition. Include these status properties in the form used in the entity layout.

Initial state

An initial state is the starting point of the Lifecycle model. The Lifecycle model always has one main initial state. A state that has sub-states also needs an initial state to determine which sub-state should be started first from the main state.

Final state

A Lifecycle model can have one or more final states. When the Lifecycle reaches the final state, activities can no longer be performed on the entity instance. A state that has sub-states can itself have a final state. When this final state is reached, the parent state becomes completed.

You can define the following properties for a state within a model.

General tab

Property	Description
Description	The provided state name is displayed at the top of the state element and can be modified. Process Experience users can select a state in the Inbox to list the activities associated with it.

Property	Description
Applicability Service	<p>The Applicability Service provides the ability to connect to a custom algorithm for recommending the most applicable activities.</p> <ol style="list-style-type: none"> 1. Select Business Process or Web Service from the Applicability Service list. 2. Click to browse and select the related business process or Web service. <ul style="list-style-type: none"> • If Business Process is selected, only the process models that implement the contract shared are displayed. Implementation of such business processes are based on Contract First Development option with the WSDL that is shared. • If Web Service is selected, only the services that implement the WSDL are considered. <p>When the Applicability Service feature is selected for the state, it becomes applicable for all the activities within the state and it is represented with an appropriate icon on the state. When the activities within the state are executed, based on the work option or exception, the Applicability Service is triggered and recommendations are made for the set of applicable follow-up activities.</p> <p>The recommendation for a follow-up activity can be of the following types:</p> <ul style="list-style-type: none"> • IsRecommended – Refers to an activity that is recommended to be executed. For example, if a customer's credit history is poor, an optional activity of verifying the previous banking transactions must be performed. • IsNotRecommended – Refers to an activity that is not recommended to be executed. However, the activity can still be manually executed, based on the discretion of the user. For example, if a patient's medical condition is clear, further diagnostic checks may not be required. • IsWarning – Refers to an activity that may be executed but with a bit of caution as it may have undesirable results. For example, if a patient's medical history mentions an allergy to anesthesia, providing it may risk the patient's condition. <p>See Using the Applicability Service in a Lifecycle model for information about how to configure the Applicability Service.</p>
State name supports translations	In Process experience, Lifecycle state translations are shown in the Lifecycle progress bar.

Transitions

When designing a Lifecycle model you need to define transition events between states and specify when they should be triggered. This ensures that the model executes exactly as you intend it to and that it is consistent with your business needs.

You can configure the transition from one state to another based on the following events:

Activity Completion Event – The transition occurs when a specific activity or all planned activities in a state are completed. Optionally, conditions on entity properties can also be specified that must evaluate to true for the transition to occur.

User Event – The transition occurs when the specified event occurs. In Process Experience, user events are displayed as action buttons in the Actions panel based on the state of the item. If a user clicks an action button, the transition occurs and the action buttons are refreshed.

You can configure entity forms on user events. When a Process Experience user performs the user event action, the corresponding form is opened so that the user can provide the required input through form properties. When the user submits the form, the captured data is saved and the action is completed. Only forms that are created on the entity where user events are defined can be configured as part of the user events.

Optionally, conditions on entity properties can also be specified. These conditions must evaluate to true and all the planned mandatory activities must be completed to display the user events as action buttons in the Actions panel.

When you add a user event, you can specify whether to enable or disable it when there are errors in the form in the application. You do this by selecting or clearing the **Allow the event even if there are errors** option.

- If **Allow the event even if there are errors** is selected, the action button is enabled and the transition to the next state occurs. By default, the check box is selected and the button is enabled even when there are errors in the form.
- If **Allow the event even if there are errors** is cleared, the action button is disabled and the user must correct the errors (for example, by filling in required fields) to enable the action button and move to the next state.

Versioning is not supported for the Lifecycle actions (user events). The latest Lifecycle actions are always shown as in the following specific cases:

- If a user event (lifecycle action) is renamed, it is no longer visible for previously-created items.
- If a user event is deleted, it is no longer visible for new items or in previously-created items.
- If the user event property called **Last Transition Event** is used in a rule condition, and if the user event is deleted, the rule fails. If the user event is renamed, the new name is used to evaluate the rule for all items and may fail for previously-created items.
- If the user event description is changed, the lifecycle action is visible with the new description for all items.

Conditional Event – The transition occurs when the condition specified is evaluated as true. These conditions can be based on the entity and related entity properties similar to what can be done in the [Rules](#) building block. See [Using the expression editor](#).

Transition properties

You can define the following properties for a transition.

General tab

Property	Description
Description	Provide a description of the transition. The description is shown above the transition line in the designer.
Transition Type	<p>Displays the type of the selected transition. You can select a different transition type from the list.</p> <ul style="list-style-type: none"> • User Event • Activity Completion Event • Conditional Event <p>When an activity is marked as mandatory, User Event and Activity Completion Event require mandatory tasks to be completed.</p>
Event name	(User Event only) Type a name for the event. If the Transition description contains any characters that are not acceptable as Event Name, those characters are replaced with underscores while auto filling Event Name.
Satisfy additional conditions	<p>Specify a condition for the transition. This is required for a conditional event and optional for other transition types. See Using the expression editor.</p> <p>Specifying a conditional expression is optional for the User Event and the Activity Completion Event.</p>
Trigger target state entry event	This option is selected by default but you can clear it if you do not want to trigger a state entry event in the target.
Trigger this event on completion of	<p>(Activity Completion Event only) Select All Activities in source state or a specific activity.</p> <ul style="list-style-type: none"> • When All Activities in source state is selected, the transition from the source state to target state is triggered only after all planned activities within the source state are completed. All Activities is selected by default. If you use a sub-process and if you terminate that sub-process, state transition will not get triggered. • If you want to trigger the transition upon completion of a

Property	Description
	specific activity, select the activity. When you select this option, a drop-down list displays a list of activities that comprise the current State. Select the activity from the list.
Set as primary transition	<p>If this option is selected, the transition will be shown as the primary transition path in the progress bar in Process Experience. Only one transition can be configured as primary for any state.</p> <p>By default, Set as primary transition is not selected for any transition that is added to a state. You must enable it manually by selecting it. A confirmation message appears if the option is changed to a different transition.</p> <p>For a state to be shown in the lifecycle progress bar in Process Experience, it must have primary transition or be in a completed state.</p>
User Event name supports translations	In Process Experience, Lifecycle user event translations are shown in the actions panel as Lifecycle actions.

Lifecycle Progress panel

The Lifecycle Progress panel enables Process Experience users to view the completed, current, and future states of an item.

If users deviate from a primary path, they can also view the alternate path from the deviation point. To use this feature, you must configure one transition for each lifecycle state as a primary transition. See [Transition properties](#).

To make the Lifecycle Progress panel available to Process Experience users you need to add it to a layout. See [Adding layouts](#).

Activities

By adding activities to a Lifecycle, you can define the tasks that will be presented to users in Process Experience or business processes to be executed. Activities can be added within a state and outside a state. If the activity is not added within a state, it can always be scheduled independent of the current state. Activities that are marked as mandatory must be executed before the state can be completed

Types of activities

You can add two types of activities you can add to a Lifecycle model.

- Human activity
- Business Process activity

Human activity

A human activity defines a task to be executed by a person. The activity can be accessed from item's Task panel in Process Experience. As part of a human activity, you can configure entity layouts and work assignments (team/work list/role/user). An activity in a Lifecycle model is by default a human activity. If no layout is linked to an activity, the activity still appears as a task in the item's Tasks panel that can be completed but that does not have a layout linked to it. If a human activity has no role, user, or team assigned, it is assigned to the user that started the item.

Only Task entity layouts can be configured as part of the human task.

Business Process activity

A Business Process activity defines a business process model to be executed as part of the entity Lifecycle. When a sub-process is started, a new process instance is created that is linked to the item. Any human tasks that are created in the process instance are visible in the Task panel of the item.

Note: Activities configured in the Lifecycle model are referred to as tasks in Process Experience.

- A Human activity corresponds to a manual task to be performed by a user.
- A Business Process activity corresponds to a system task and does not appear in the list of tasks. However, the human activities configured in the business process do appear as human tasks in the list of tasks.

Note: When you add a Business Process activity to a lifecycle, be sure that there are no sub-cases (Classic cases) added as activities in the Business Process Model.

- Task entity layouts can also be configured for human activities in a business process, similar to lifecycle human activities.

Activity connectors

Activity connectors enable you to plan specific activities. Careful planning provides Process Experience users with the capability to handle exception situations that are not specifically defined as part of the Lifecycle definition.

For example, users can be empowered to handle exceptions by providing the following functions:

- Adding ad hoc tasks.
- Adding relevant discretionary tasks based on the state of a task.
- Planning future tasks, for example assign tasks when an item transitions to a subsequent state.
- Adding related tasks for a specific task that is assigned to the user.

You make these functions available by using the appropriate connector to link two activities.

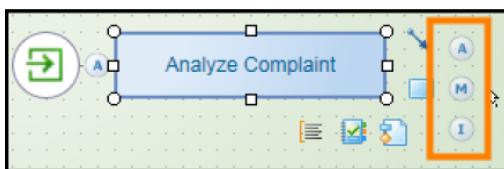
As you hover over the objects to which you can connect, small letters appear next to each object.

After designing the Lifecycle model you must set the properties of the connectors to define when they should be triggered. This ensures that the Lifecycle model executes exactly as you intend it to and is in line with your business needs.

During execution of an activity or when it is completed, you can plan follow-up activities. You can also plan intermediate follow-up activities but only during the execution of an activity and not when an activity is completed. For example, during approval of an order, a Legal Counselor who wants a second opinion plans an extra activity and assigns this activity to a colleague.

Only activities that follow the current activity or free follow-up activities can be planned. Activities that are in another state can also be planned, as long as they are connected to the current activity with a follow up or are free follow-up activities. These activities will be scheduled when the other state becomes active. All follow-up activities, except activities that have an incoming intermediate follow-up connector, can be planned by the end-user after completion of an activity.

As you position the pointing device over the objects to which you can connect, small letters appear next to each object.



These letters represent the types of connectors that are automatically added when the object is placed.

A connector can be drawn to another activity from a state entry, document received event, or any activity. This follow-up activity can be part of the same state but can also be part of any other state. Possible connector types are described in the following sections.

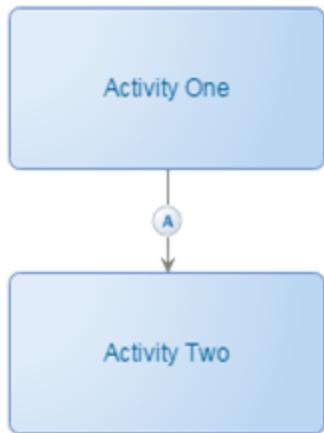
Automatic follow-up connector

A user who completes an activity can plan the next activity (from the item, as part of completing an activity). By default each activity that is configured as an automatic follow-up activity (from the activity that is being completed) is selected. A user can choose to remove an automatic activity (except in the case of a mandatory activity). Although the work assignment (such as role or user) and due date can be defined as part of the Lifecycle model, the user who plans the activity can specify or change the work assignment (the who), when (due date), and (in the case of a human activity), the priority in which the next activity should be done.

Automatic follow-up connectors can be drawn from:

- State entry - The follow-up activity is automatically planned when the state is entered.
- Business Process activity - The follow-up activity is automatically planned after the Business Process activity is completed.
- Human activity - The follow-up activity is planned after the human activity is completed.
- Document received event - The follow-up activity is planned after the configured document is received

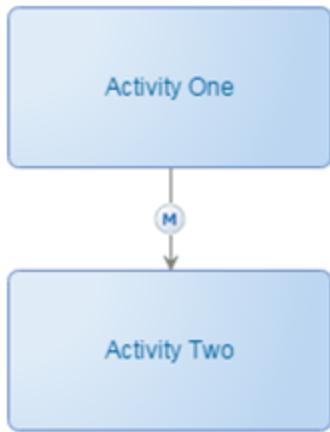
After a state entry, document received event, or business process activity, only automatic (not manual or intermediate) follow-up connectors can be drawn.



When Activity One is connected to Activity Two using an automatic connector, completion of Activity One triggers execution of Activity Two.

Manual follow-up connector

A manual follow-up connector can only be drawn after a human activity. When the human activity is completed from the item's task panel, the user who completed the activity can decide whether the manual follow-up activity should be planned, by whom, when (due date), and with which priority.

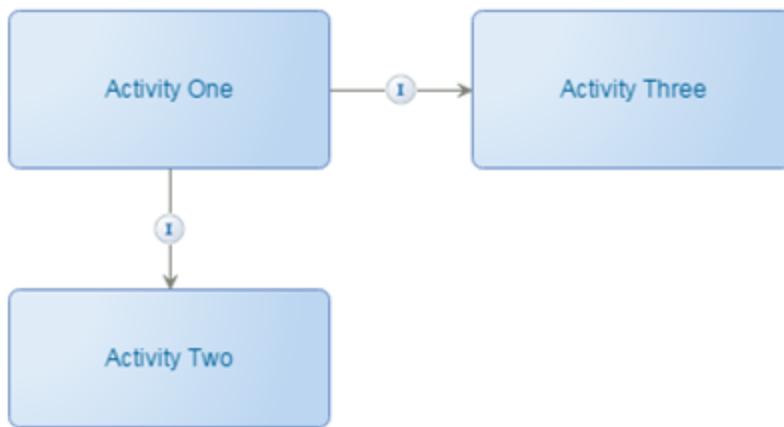


When Activity One and Activity Two are connected through a manual connector, completion of Activity One provides an option to configure and add Activity Two, if relevant.

Intermediate follow-up connector

When working on an activity, a user can decide whether to plan an intermediate follow-up activity. If an intermediate activity is appropriate, the current activity is suspended from the item's task list and resumes again after intermediate follow-up activity is completed (except when the intermediate follow-up triggered a state transition).

All the activities with incoming intermediate connectors can be added through the Add related tasks option in the task list.



When Activity One is selected in the task list, Activity Two and Activity Three are available as related tasks.

Activities without connectors

Activities without any incoming connectors can be added to a state or outside a state. These activities can be added or planned by the users while completing an activity or through the Add Tasks option in Process Experience.

Activity planning guidelines

During execution of an activity or when it is completed, you can plan follow-up activities. You can also plan intermediate follow-up activities but only during the execution of an activity and not when an activity is completed. For example, during approval of an order, a Legal Counselor who wants a second opinion plans an extra activity and schedules this activity to a colleague.

Only activities that follow the current activity or free follow-up activities can be planned. Activities that are in another state can also be planned, as long as they are connected to the current activity with a follow-up or are free follow-up activities. These activities will be scheduled when the other state becomes active. All follow-up activities, except activities that have an incoming intermediate follow-up connector, can be planned by the end-user after completion of an activity.

Activity properties

You can define the following properties for an activity.

General tab

Property	Description
Description	Provide a description of the activity.
Priority	<p>Specify the priority for the activity by selecting Static or Read from property options from the list.</p> <p>Static - You can select any one of the priority levels: High, Normal or Low</p> <p>Read from property - If you want to dynamically set the execution priority using entity property, click  (Lookup) to provide the property.</p> <p>See Configuring expressions > Configuring an expression to set a property.</p> <p>By default, the priority level property set at Lifecycle model level will be inherited by all the activities of the Lifecycle model. You can override the default priority value at the activity level using this property.</p>
This activity must be performed (required activity)	<p>Select This activity must be performed (required activity) to ensure that execution of the current activity is enforced. By default, the activities of type automatic follow-up are planned during Lifecycle execution. However, Process Experience users still have an option to remove such activities from execution through the follow-up view displayed on completion of task. Setting this property disables the removal and forces users to perform the action to complete it.</p> <p>Note:</p> <ul style="list-style-type: none"> • This property is applicable only for activities of type Automatic follow-up. • A State transition does not occur if there are any pending activities with this property enabled within the current state.

Layout Model tab

Property	Description
Layout	Provides a lookup to a list of all the layouts created on the child Task entity from which you can make a selection. This layout will be launched on opening of the task item in Process Experience.

Duration tab

Property	Description
Business Calendar	<p>Business Calendar enables you to associate the timely business activities with the Lifecycle model. Select the required Business Calendar by browsing the list from the Select a Business Calendar dialog box. If you want to continue with the default calendar, that is 24*7, do not provide any changes.</p> <p>By default, the business calendar configured at the Lifecycle model level will be applicable for all its activities. You can override the default business calendar at an activity level using this property.</p> <p>See the Process Platform product documentation for information about creating and using business calendars.</p>
Duration Type	<p>Specify the duration (in Days, Hours and Minutes) during which a user needs to complete an activity. You can select Static Time or Read from property.</p> <ul style="list-style-type: none"> If Static Time is selected, specify the number of days, hours, and minutes. If Read from property is selected, you can specify the duration dynamically by providing an entity property. Click  (Lookup) to select the property. <p>See Configuring expressions > Configuring an expression to set a property.</p> <p>By default, the duration configured for the Lifecycle model will be applicable for all its activities. You can override the default duration at an activity level using this property.</p>

Work Assignment tab

Tab	Property	Description
Work Assignment	Assignee Type	<p>Based on your business requirements, you may have a single user, multiple users, or teams working on the task item. Therefore, it is important to select an assignee type to clearly indicate who must work on the current task item. Click to add the type of assignment and to add users, a team, a role, or a worklist so that you can assign the activities of the Lifecycle as follows:</p> <p>Team - The users in the specified team who must work on this Lifecycle or activity.</p> <p>Role - The set of users with this role who must work on</p>

Tab	Property	Description
		<p>this activity.</p> <p>Worklist - The list of teams associated with the work list.</p> <p>User - The specific user. When you select this value, you must create or select an entity property that contains the value of a role, team, or user that specifies the users. See Configuring expressions > Configuring an expression to set a property.</p>

Work Distribution tab

Work distribution is a custom algorithm that you can define to distribute the work among the users. Select one of the following work distribution algorithms:

- Inherit from Lifecycle - Select this option to inherit the work distribution configurations from the Lifecycle model level.
- Static Dispatch Algorithm - Click  (Lookup) to select the algorithm to be used to distribute work.
- Dispatch algorithm from property - Click  (Lookup) to select the property to be used to distribute work.

See [Configuring expressions > Configuring an expression to set a property](#).

For information about creating and working with dispatch algorithms, see [Working with Dispatch Algorithms](#).

Business Process properties

You can define the following properties for a Business Process activity.

General tab

Property	Description
Description	A description of the business process model.
Business Process Model	The name of the business process model. Select a business process model from the lookup provided. Entity item information (Item ID) will be available as a property (rootEntityInstanceId) of the process instance in the business process.
Wait until Sub Process is Finished	Whether to wait until the sub process completes before executing the next process.
Priority	When you execute multiple business processes, corresponding process instances are created. If you want to specify the execution

Property	Description
	<p>priority for these process instances select priority level. Using the Priority list, you can select a priority level to provide the priority of the execution.</p> <p>The following options are available:</p> <ul style="list-style-type: none"> • Same as Sub Process - The priority is same as defined in the selected business process. • Other - To specify the execution priority that is different to the sub process, select one of the following: <ul style="list-style-type: none"> ◦ Static - Select one of the priority levels: Highest, High, Normal, Low, and Lowest. ◦ Read from property - Select this option if you want to dynamically set the execution priority via an entity property. See Configuring expressions > Configuring an expression to set a property. <p>Note: If the priority is not set, the items are executed in FIFO (First-In, First-Out) order. During the execution, the model with High priority takes precedence over the models with lower priority levels.</p>
InstanceId	<p>Opens the expression editor so that you can look for an InstanceId. See Configuring expressions > Configuring an expression to set a property</p>

Document Received properties

You can define the following properties for a Document Received event.

General tab

Property	Description
Name	The name of the document received event.
Description	A description of the event.
Group	Displays a list of groups from the Content building block. Select a group from the list.

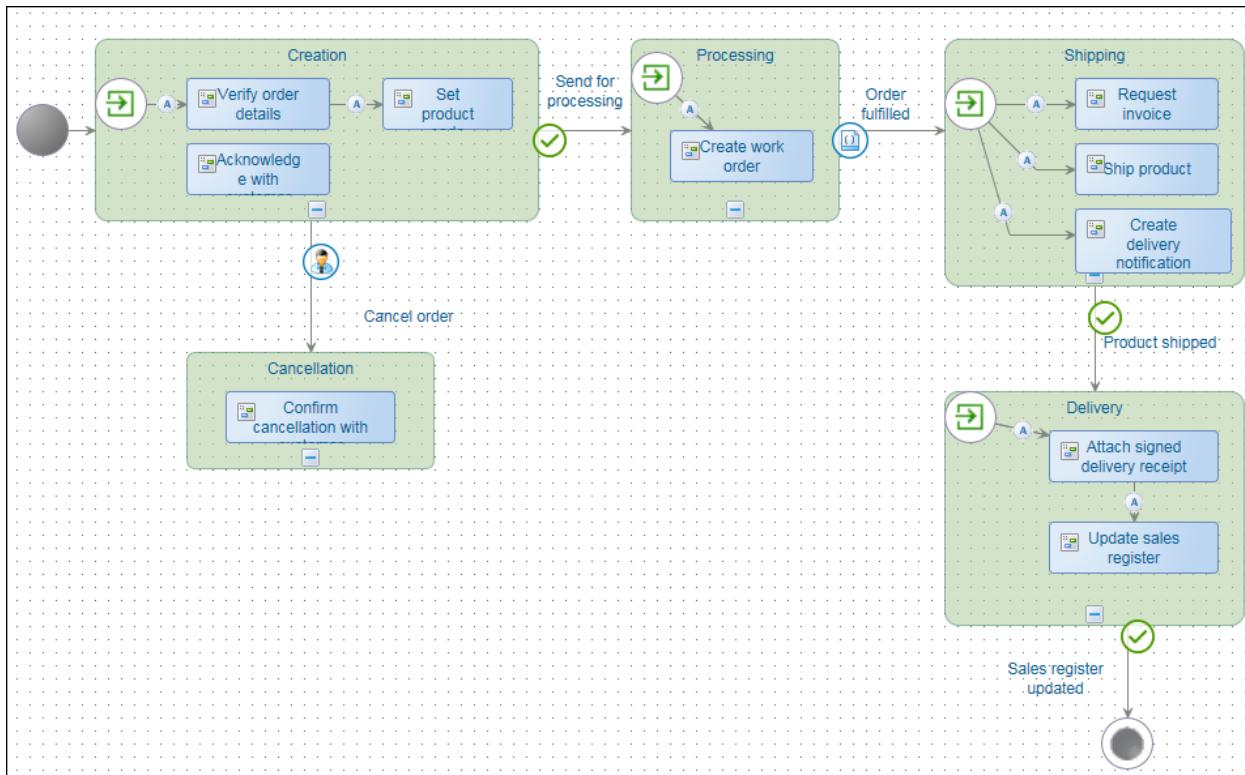
Lifecycle examples

This section provides sample Lifecycle models that may be helpful as references for building your own models

Order management

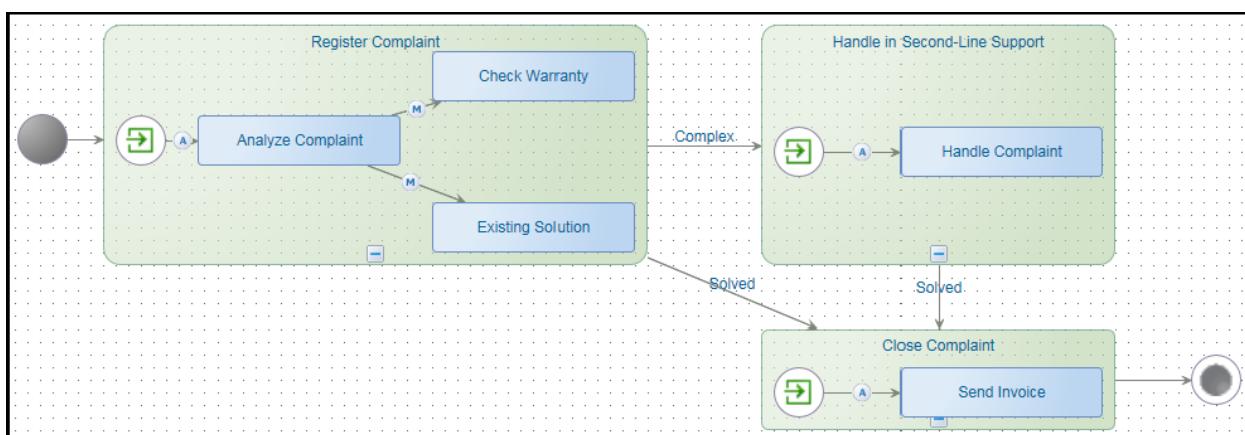
An example of an order management Lifecycle follows.

Note: The model is very simple and is intended for illustration only. It is formatted to enable viewing it within in the documentation. In an actual production environment, it may make more sense to arrange the states horizontally.



Complaint handling

An example of a complaint handling Lifecycle model follows.



Adding a lifecycle

You can use the Lifecycle building block to specify how an entity can progress through various stages and to define how it responds to events and tasks (called activities in a Lifecycle model).

When you add a Lifecycle building block to an entity, a building block called Task list, a child entity called LifecycleTask, and a Rule building block are automatically created. By default, the child entity includes Identity, Task, and Relationship building blocks. The Rule building block is required to execute any conditional events configured in the Lifecycle.

See [Configuring the task entity](#) for information about additional building blocks that you can add to the Tasks child entity.

To add the Lifecycle building block:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Lifecycle**.
3. Click **Add**.

Note: A building block called Task list is automatically added to an entity when you add a Lifecycle building block.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

4. Click  (Save).
5. Click **Configure**. The Lifecycle Designer opens.
6. Define the properties of the Lifecycle model. See [Lifecycle model properties](#).
7. Construct the model. See [Creating a Lifecycle model](#).
8. Configure task through the Task list building block. See [Configuring the task entity](#).

Note: When you add a Lifecycle building block, a project called Default Case Management Event Types is automatically created in your workspace. You do not need to do anything with this project. It is for internal use only.



In order to work with Lifecycle, be sure to run the Business Process Management and Notification service containers inside TomEE.

To run the Business Process Management and Notification service containers inside TomEE:

1. On the Service container properties page, select the **General** tab.
2. Select **Assign OS Process** and then select **TomEE OS process** from the list.
3. Restart the service container.

Lifecycle building block properties

The following properties are automatically added to the Lifecycle building block. You can add these properties to forms and lists and use them as part of expressions for defining rules. For example, you can add State to a form and rename the label as Status.

In expressions, these properties must be referred to using the fully qualified name. For example, the fully qualified name for the State property is item.Lifecycle.CurrentState. See [Configuring expressions > Configuring an expression to set a property](#)

Property display name	Property name	How to access	Description
Due date	DueDate	item.Lifecycle.DueDate?	The due date of the case instance. This can be used in business rules.
Instance identifier	InstanceIdentifier	item.Lifecycle.InstanceIdentifier	A unique identifier that refers to the instance of the Lifecycle model.
Instance status	InstanceState	item.Lifecycle.InstanceStatus	The current status of the Lifecycle. This can be used in business rules or to identify the various statuses of a Lifecycle from an entity.
Last performed task	PriorActivity	item.Lifecycle.PriorActivity	The most recent activity that was performed on the Lifecycle model.
Last transition event	PriorEvent	item.Lifecycle.PriorEvent	The most recent user event that was performed on the Lifecycle model.

Property display name	Property name	How to access	Description
Parent state	ParentState	item.Lifecycle.ParentState	The state of the immediate parent of the current state of the Lifecycle. If there is no parent state, the current state is registered as the parent state.
Previous state	PreviousState	item.Lifecycle.PreviousState	The previous state of the Lifecycle model.
Previous state ID	PreviousStateId	item.Lifecycle.PreviousStateId	The previous state's ID. This value is updated when the PreviousState property is updated.
State	CurrentState	item.Lifecycle.CurrentState	The current state of the Lifecycle.
State ID	CurrentStateId	item.Lifecycle.CurrentStateId	Contains current state's ID. Like the CurrentState property, this value is updated from the Case engine.
Top-level state	RootState	item.Lifecycle.RootState	The root state in the state hierarchy. If the current state of the Lifecycle is not a sub state, the current state is registered as the root state.

Lifecycle building block actions

Add Tasks is a standard action for the Lifecycle building block. It gives Process Experience users the ability to plan tasks. See the *Process Experience User's Guide* for a description of each available task. Apart from the standard actions, all the transition types of user event are exposed as action buttons.

Access to these actions can be restricted through security policies that are defined using the Security building block (see [Configuring security](#)). Like other action buttons, the visibility of these actions can be handled through the action bar presentation (see [Creating an action bar](#)). The **Add Tasks** action is also available as part of the Tasks panel that can be configured in an entity layout.

Creating a Lifecycle model

By creating a Lifecycle model, you graphically depict how an entity will move through a business process. You do this using the Lifecycle Designer.

Tip: It is a good idea to diagram the business process on paper and have it reviewed before you begin configuring it in the Lifecycle Designer.

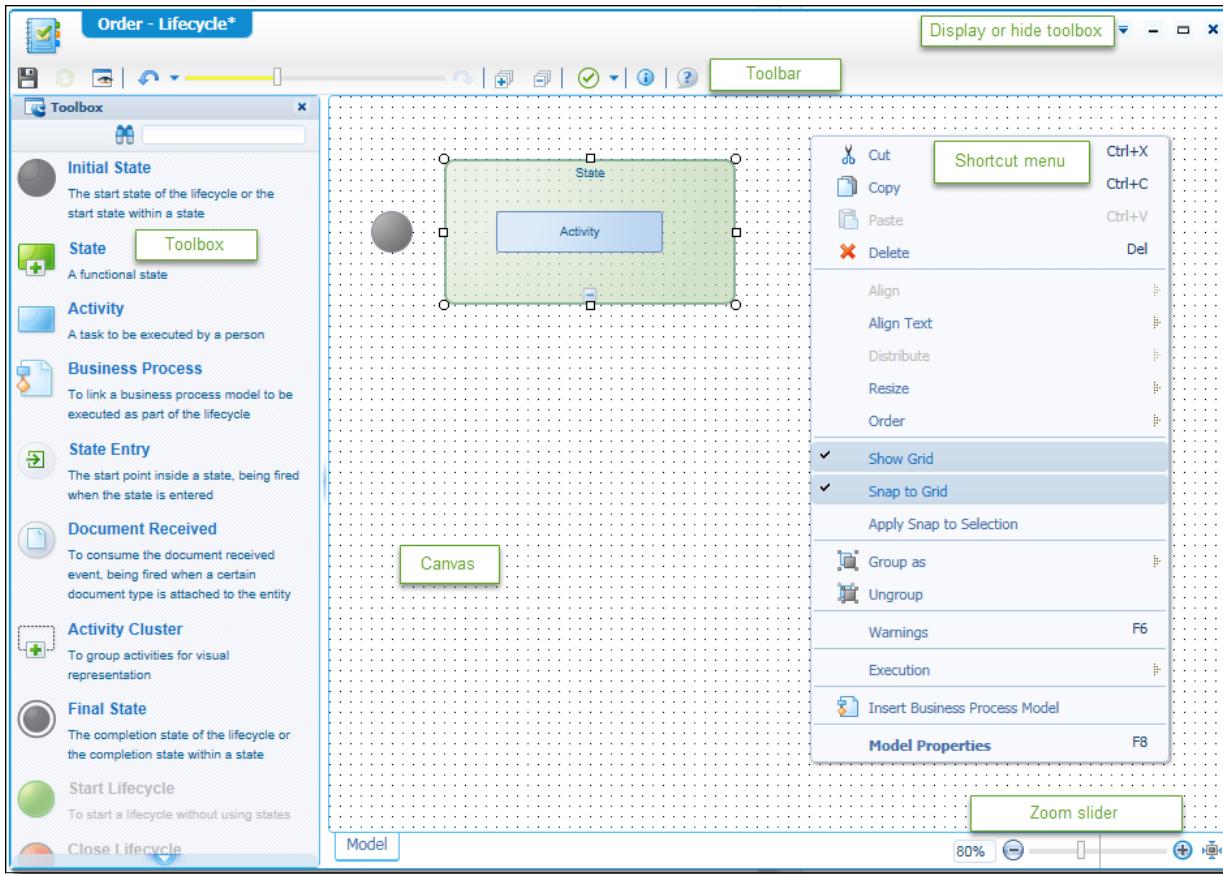
Before you begin, add the Lifecycle building block to the entity. See [Adding a lifecycle](#).

To design a Lifecycle model:

1. In **Workspace Documents**, open the entity and select the Lifecycle building block.
2. In the **Lifecycle Details** pane, click **Configure**. The Lifecycle Designer is displayed.
3. Build your model by dragging the required components to the canvas and configure their properties. See the following topics:
 - [Lifecycle Designer](#)
 - [Constructs](#)
 - [Configuring the task entity](#)
 - [Configuring expressions](#)
 - [Validating and building the model](#)

Lifecycle Designer

The following sections describe features of the Lifecycle Designer that can help you efficiently build and format a model.



Toolbar

The toolbar provides icons that you can use to perform the following functions:

- Save your model. It is best practice to save frequently.
- Refresh the display.
- Display a preview of how the model will appear when printed.
- Undo unsaved changes. You can display a history of unsaved changes by clicking the down arrow.
- Redo. Using the slider, you can redo one or more or all changes.
- Expand or collapse all groups that were added to an activity cluster.
- Select the default option for a follow-up connector.
- Validate the model.

Toolbox

A toolbar appears in the left pane of the Lifecycle Designer. It contains icons (called constructs) that you can drag to the canvas to design your model. It also provides a search box that you can use to find a particular construct.

Zoom slider

You can use the slider at the bottom right corner of the Lifecycle Designer to zoom in or out on your model so that you can see more or less detail. Click the icon to the right of the slider to fit your model in the window.



Selection methods

There are two ways to select objects for a model.

- Drag the pointing device to lasso the objects that you want to select.
- Move the pointing device over an object until it appears as a four-headed arrow and click.

Shortcuts

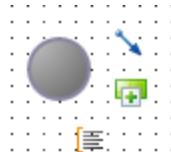
If you right-click the canvas, shortcuts are available to help you with the design process.

Shortcut	Description
Cut, Copy, Paste, Delete	Cuts, copies, pastes or deletes the selected model objects.
Align, Align Text > Top, Middle, Bottom, Left, Center, Right	Aligns selected objects or text. The components are aligned with the most recently selected object.
Distribute	Evenly distributes, increases, or decreases horizontal or vertical space for selected objects.
Resize	Makes the selected objects the same size, width, or height, or designates the selected object as the default size.
Order	Sends the selected objects to the back or front.
Show Grid	Shows or hides the grid.
Snap to Grid	Snaps all objects to the grid.
Apply Snap to Selection	Snaps the selected objects to the grid.
Group as, Ungroup	Groups selected objects as a State or Activity cluster, or to ungroup grouped objects.
Warnings	Validates the model, after prompting you for confirmation, and displays a list of warnings.
Execution	Validates and builds the model and shows the generated output.
Insert Business	Inserts a business process model that you select in the Lifecycle.

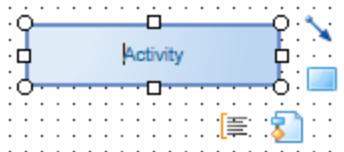
Shortcut	Description
Process Model	
Lifecycle Design Properties	Opens the Properties sheet for the model at the bottom of the window. See Lifecycle model properties .

Tip: The Model Designer provides an simplified way to draw constructs. When you drag a construct to the canvas, all of the possible constructs that can be used with it are displayed.

The following image shows the constructs that can be used immediately after the Initial State construct.



The following image shows the constructs that can be used immediately after the Activity construct.



Lifecycle model properties

A Lifecycle contains properties that describe specific characteristics of the entire model or of a state, activity, or transition within the model. You can either configure properties as you create each element or you can create the elements and then go back and define their properties.

To view and set the properties of a model:

- For model properties, right-click the canvas and select **Model Properties**.

To view and set the properties of a state, activity, or transition:

- Select the element, right-click the four-headed arrow, and then click **Properties**.

You can define the following properties for the model.

General tab

Property	Description
Root State Name	All the follow-up activities defined in the model that are not associated with a specific user-defined state are considered to be part of this state. This is an editable field and you can modify the default name provided. When Process Experience users perform an

Property	Description
Applicability Service	<p>Add tasks action on an item, they can select a state name to list the follow-up activities that are not associated with any specific user-defined state. See the Add tasks functionality section in the <i>Process Experience User's Guide</i>.</p> <p>The Applicability Service provides the ability to connect to a custom algorithm for recommending the most applicable activities.</p> <ol style="list-style-type: none"> 1. Select Business Process or Web Service from the Applicability Service list. 2. Click to browse and select the related business process or Web service. <ul style="list-style-type: none"> • If Business Process is selected, only the process models that implement the contract shared are displayed. Implementation of such business processes are based on Contract First Development option with the WSDL that is shared. • If Web Service is selected, only the services that implement the WSDL are considered. <p>When the Applicability Service feature is selected for the state, it becomes applicable for all the activities within the state. When the activities within the state are executed, based on the work option or exception, the Applicability Service is triggered and recommendations are made for the set of applicable follow-up activities.</p> <p>The recommendation for a follow-up activity can be of the following types:</p> <ul style="list-style-type: none"> • IsRecommended – Refers to an activity that is recommended to be executed. For example, if a customer's credit history is poor, an optional activity of verifying the previous banking transactions must be performed. • IsNotRecommended – Refers to an activity that is not recommended to be executed. However, the activity can still be manually executed, based on the discretion of the user. For example, if a patient's medical condition is clear, further diagnostic checks may not be required. • IsWarning – Refers to an activity that may be executed but with a bit of caution as it may have undesirable results. For example, if a patient's medical history mentions an allergy to anesthesia, providing it may risk the patient's condition. <p>See Using the Applicability Service in a Lifecycle model for information about how to configure the Applicability Service.</p>

Duration tab

Property	Description
Business Calendar	<p>Business Calendar enables you to associate business activities with the Lifecycle model. Select the required Business Calendar from the available list and attach to the Lifecycle model. If you want to continue with the default calendar, that is 24*7, do not provide any changes.</p> <p>See the Process Platform product documentation for information about creating and using business calendars.</p>
Duration Type	<p>Specify the duration (in Days, Hours and Minutes) during which a user needs to complete an activity. You can select Static Time or Read from property.</p> <ul style="list-style-type: none"> If Static Time is selected, specify the number of days, hours, and minutes. If Read from property is selected, you can specify the duration dynamically by providing an entity property. Click  (Lookup) to provide the entity property. See Configuring expressions > Configuring an expression to set a property. The property type must be Duration. <p>By default, the duration configured for the Lifecycle model will be applicable for all its activities. You can override the default duration at an activity level using this property.</p>

Work Assignment tab

Property	Description
Assignee Type	<p>Based on your business requirements, you may have a single user, multiple users, or teams working on the entity item. Therefore, it is important to select an assignee type to clearly indicate who must work on the current entity item. Click to add the type of assignment and to add users or teams so that you can assign them the activities of the Lifecycle as follows:</p> <ul style="list-style-type: none"> Team - The users in the specified team who must work on this Lifecycle or activity. Role - The set of users with this role who must work on this Lifecycle or activity. Worklist - The list of tasks on which the teams are working. User - The specific user. When you select this option, you must create or select an entity property that contains the value of a role, team, or user that specifies the users. See Configuring expressions > Configuring an expression to set a property.

Work Distribution tab

Work distribution is a custom algorithm that you must define to distribute the work among the users. Select one of the following work distribution algorithms:

- Use system default (Round robin) - The default settings of the system determine the work distribution in a round robin fashion.
- Static Dispatch Algorithm - Click  (Lookup) to select the algorithm to be used to distribute work.
- Dispatch algorithm from property - Click  (Lookup) to select the property to be used to distribute work.

See [Configuring expressions > Configuring an expression to set a property](#)

See the information about working with dispatch algorithms in the Process Platform Product documentation.

Constructs

The Lifecycle Designer toolbox provides icons that represent elements (called constructs) that you use to design a Lifecycle model. The following table describes each construct and shows the additional elements you can add to it when it is selected on the canvas. The available elements depend on the construct that is selected.

Icon	Construct	Description	Possible elements that can be added from the construct
	Initial State	The start state of the Lifecycle model or the start state within a state.	State transition State Text Annotation
	State	A functional state. See	State transition State Final State Text Annotation
	Activity	A task to be executed by a person. See	Connector <ul style="list-style-type: none"> ▪ Automatic follow-up connector ▪ Manual follow-up connector ▪ Intermediate follow-up connector Activity Business Process <ul style="list-style-type: none"> ▪ Automatic follow-up connector ▪ Manual follow-up connector ▪ Intermediate follow-up connector

Icon	Construct	Description	Possible elements that can be added from the construct
			Close Lifecycle Text Annotation
	Business Process	A business process model to be executed as part of the Lifecycle.	Automatic follow-up connector Activity Business Process Close Lifecycle Text Annotation
	State Entry	The start point that is triggered inside a state when the state is entered.	Automatic follow-up connector Activity Business Process Text Annotation
	Document Received	<p>The document received event that is triggered when a certain document is attached to the item.</p> <p>The property sheet of the Document Received Event lists the groups that are created on the Content building Block of the entity. Any of the groups listed can be selected as the group for the Document Received event.</p> <p>In Process Experience, when a document is uploaded to the specified group, the Document Received event is triggered in the Lifecycle model where further actions can be configured.</p>	Automatic follow-up connector Activity Business Process Text Annotation
	Activity Cluster	Groups activities for visual representation.	Text Annotation

Icon	Construct	Description	Possible elements that can be added from the construct
	Final State	The completion state of the Lifecycle or a state within the Lifecycle.	Text Annotation
	Start Lifecycle	The start point of a lifecycle without using states.	Automatic follow-up connector Activity Business Process Text Annotation
	Close Lifecycle	The completion of a Lifecycle without states.	Text Annotation
	Text	Text with a specified border and background color.	
	Text Annotation	An annotation that can be linked to a shape.	Informal Relationship
	Transparent Text	Transparent Text without borders and a background.	

Configuring the task entity

When you add a Lifecycle building block or Activity flow building block to an entity, a child entity called LifecycleTask is automatically created. This can be configured using the **Configure task** option on the Task list building block.

Important: By default, the child entity also creates Identity and Relationship building blocks and the Activity flow task building is also added by default if the parent entity has an Activity flow building block. No delete option is available for the Identity and Activity flow building blocks. However, you should not delete any of the implicitly added relationships and building blocks.

- The Task building block adds standard task actions and properties to the Task entity. These can be used in the other building blocks of the entity.
- The Relationship building block called ParentEntity defines a relationship with the parent entity. This enables the use of properties from parent entity in other building blocks of the Task entity.

You can add Action Bar, Business Workspace, Content, Discussion, Email, Email Template, File, Form, History, Layout, List, Mobile App, Property, Relationship, Rule, Security, Title, Tracking, and Web service building blocks to the child entity.

Task actions

The Task entity provides standard task actions. For example, Process Experience users can claim tasks, start the claimed or assigned tasks, pause the tasks when required, and so on. Based on the permissions, users can perform basic operations such as, claim task, start task, pause task, complete task, and so on. A team leader or work list manager can perform additional operations, such as skipping a task, suspending a task, forwarding a task to another work list or team, and so on. Layouts defined on the child entity can be used to display Lifecycle tasks in Process Experience. See the *Process Experience User's Guide* for a description of each available task.

Task properties

The Task entity also provides properties that can be used in building blocks, such as forms and lists, for the Task child entity. These properties are supplied by the Inbox Task Management application. See [Defining security for Inbox Task Management](#)

Tip: When using these properties in a building block (such as a form), you can assign them labels that will be meaningful for your application.

Note: Calculated properties of the LifecycleTask entity, such as Task Owner Name, Status, and Target Name, are not available for filtering lists, searching, and sorting.

Any properties other than those shown in the following list are for internal use only.

Property	Description
Acl Id	The access control identifier that stores the security related information of the task.
Activity Id	The internal ID of the activity.
Application Url	The URL of the application or the user interface that needs to be opened.
Assign On	The date when the task was assigned.
Callback Info	The information about the caller (BPM or Case) of task.
Completed By	The user who completed the task.
Completed By Name	The name of the user who completed the task.
Completion Date	The date the task was completed.
Delegated To	The user to whom the task is delegated.
Delegated To Name	The name of user to whom the task was delegated.
Delete Flag	Whether the task was deleted. This is used for legacy task models.
Delivery Date	The date when the task was created.

Property	Description
Dependent	The task that must be completed.
Due Date	<p>The date when the task is due.</p> <p>Note: When the lifecycle task due date elapses, the color of the task due date field in the task panel does not change until the page is refreshed.</p>
Email Model Id	Unique identifier of the Email model.
EntityInstanceId	The item id of main\root entity instance.
Entity Layout Id	The runtime document id of the entity layout that is configured to the task.
Exclusion List	DN of the user who should be excluded from executing the task.
Human Task Id	The internal ID of a human task.
Inbox Model Id	The unique identifier of the Inbox model.
Is Priority Fixed	Whether the priority of the task can be changed.
Lock By	The user to whom the task is locked. This is used for legacy task models.
Logger Context	Detailed information about the context of an application. It is propagated across all the activities or transactions involved in that application.
Message Header	<p>The following information pertaining to task when it is forwarded to another user:</p> <p>SENDER - The DN of the sender who sent the message initially.</p> <p>RECEIVER- The DN of the receiver who received the message initially.</p> <p>MEMO - The message that was forwarded.</p>
Parent Source Instance Id	The internal ID of the parent item.
Parent Task Id	The internal ID of the parent task.
Parent Worklist	If the task was forwarded, the worklist that stores the previous target container details.
Priority	The priority of the task. It is set between 0 to 5 (low to high).
Read State	Whether the task is read only. This is used for legacy task models.
Root Case Instance Id	The Instance Id of the source (BPM/Case) of the task.
Sender	The user who sent the task.

Property	Description
Sender Name	The name of user who sent the task.
Source Instance Id	The internal ID of the item.
Source name	The name of the source where the task originated.
Source Type	The type of the source where the task originated.(for example, BPM/CASE/Web service).
Start Date	The date when the task was started.
Started On	The date when work on the task was started.
State	The current state of the task.
Subject	The subject of the task.
Target Name	The target name where the task is sent.
Target Type	The assignment type of the task.
Task Data	The application data and custom data of the task.
Task Entity Instance Id	The item id of the task entity instance.
Task Information	Task meta data information stored in XML format.
Task Owner	The owner of the task.
Task Owner Name	The name of the user who owns the task.
Time Taken	The amount of time it took to complete the task.

Configuring task actions using action bars

Various actions of the child entity are made available for adding to an action bar. You can add an action bar to the Task child entity. The action bar specifies which actions will be available to Process Experience users.

Important: Do not include a Delete button in the action bar because deletion should not be allowed for a created task.

To add layouts for various types of tasks:

1. Create the required action bars. See [Creating an action bar](#).
2. Create the forms that will be used to open tasks. You can use properties from the Task entity and the Lifecycle parent entity. See [Adding forms](#).
3. Create a layout that includes **Breadcrumb**, **Action**, and **Form** panels.
4. In the **Form** panel, select the form to use for one or more activities. See [Adding layouts](#).
5. In the Lifecycle Designer, select the layout to use as a property of each activity.

Associating an activity with a layout

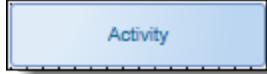
When you add a Lifecycle building block to an entity, the LifecycleTask child entity is created automatically. Only the layouts defined on this child entity are available while selecting layouts for Lifecycle activities.

By creating different forms for each activity, including each form in a Form panel in a layout, and then selecting the layout in the layout properties, you can specify which form should be used to work with each activity.

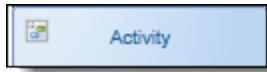
To associate a Lifecycle activity with a layout:

1. Open the Lifecycle activity property sheet and select the **Layout Model** tab.
2. Select **Layout Zoom** to list the layouts created on the LifecycleTask entity.
3. Select the layout to associate with the Lifecycle activity.

An activity that is not associated with a layout appears as follows in Lifecycle Designer:



An activity that is associated with a layout appears as follows in Lifecycle Designer:



Setting permissions for tasks

Typically, not all Process Experience users will be authorized to perform all tasks. For example, your organization may want only managers to be able to forward tasks. To do this, you add the Security building block to the Task entity.

Note: If the Security building block is added to the Task entity, it is mandatory to explicitly define the permissions on the Lifecycle Task (child) entity for all required actions. If the Security building block is not added, all the permissions are granted by default.

To set permissions for tasks:

1. Open the Task child entity.
2. Click **Add**.
3. In the list of building blocks, select **Security > Add**.
4. In the **Security** details panel, click **Configure**.

Use the Security Editor to grant permissions on task actions to roles. In the left panel, add the roles to which you want to grant certain permissions. In the right panel, specify the tasks actions that are allowed for a selected role. See [Configuring security](#).

Configuring expressions

There are various locations in the Lifecycle Designer where the expression editor is available for selecting properties or for configuring an expression as explained in the following sections.

Configuring conditional expressions

You can define conditional expressions for Lifecycle transitions. See [Using the expression editor](#).

Configuring an expression to set a property

You can use entity properties to define various properties of Lifecycle objects such as duration, assignee, work distribution, and so forth. In Process Experience, these values will be set with the actual property values so that Lifecycle execution behavior can be regulated.

You can specify the entity properties by using the expression editor. For this purpose, the expression editor is not used to create an expression, but it is used to validate whether the specified property exists.

To specify the entity property, type the fully qualified property name for a property defined in the entity. See [Expression language > Properties](#).

An example is item.Properties.Duration, as shown in the following screenshot.



Note: When you configure the expression using Expression editor, the lifecycle is saved automatically.

Following are the expected values for the configured entity property using expressions for the various lifecycle properties.

Property name	Expected property value or type
User	User distinguished name (DN)

Property name	Expected property value or type
	<p>For example:</p> <pre>cn=pjohn,cn=organizational users,o=system,cn=cordys,cn=defaultInst,o=opentext.net</pre>
Role	<p>Role distinguished name (DN).</p> <p>For example :</p> <pre>cn=Service Agent,cn=organizational roles,o=system,cn=cordys,cn=defaultInst,o=lab.opentext.com</pre>
Organizational unit	Name of the organizational unit.
Work List	Name of the work list.
Duration	The property type must be Duration. See Adding properties .
Priority	<p>A value from 1 to 5</p> <p>5 - Highest priority</p> <p>1 - Lowest priority</p>

Validating and building the model

You need to validate and build the model to check for any warning and errors in the design. On validate it prompts you for confirmation, and displays a list of warnings.

On successful validation you can publish the entity which also publishes the Lifecycle changes. Published changes of the lifecycle model will be applicable only for newly created entity items and not for previously created and published entity items.

Using the Applicability Service in a Lifecycle model

While working on a Lifecycle model, a user may need to choose the most applicable subset from the next set of follow-up activities by considering all the possible options and exceptions available for executing a task.

The Applicability Service enables users to consider all the possible options and exceptions by implementing the required business logic on a Web service. When the Applicability Service feature is selected for a state, it becomes applicable for all the activities within the state. When the activities within the state are executed, the Applicability Service is triggered, based on the work option or exception, and recommendations are made for the set of applicable follow-up activities.

Recommendation types

The recommendation for a follow-up activity can be of the following types:

- `isRecommended` – An activity that is recommended to be executed. For example, if a customer's credit history is poor, an optional activity of verifying the previous banking transactions must be performed.
- `isNotRecommended` – An activity that is not recommended to be executed. However, the activity can still be manually executed, based on the discretion of the user. For example, if a patient's medical condition is clear, further diagnostic checks may not be required.
- `isWarning` – An activity that can be executed but with a bit of caution as it may have undesirable results. For example, if a patient's medical history mentions an allergy to anesthesia, providing it may risk the patient's condition.

Implementing the Applicability Service

The applicability logic can be implemented as a Web service or a Business Process Model (BPM) directly. A web service can be implemented using a Java class, a decision table, or a business process. This topic explains the process of implementing the Applicability logic using a business process.

Scenario

Consider a Lifecycle model that is designed to handle providing facilities for passengers for trains that are delayed due to technical reasons. In such a situation, the administration needs to provide facilities to the train passengers based on the current condition in a station. The process is handled by a Lifecycle model containing a set of defined activities.

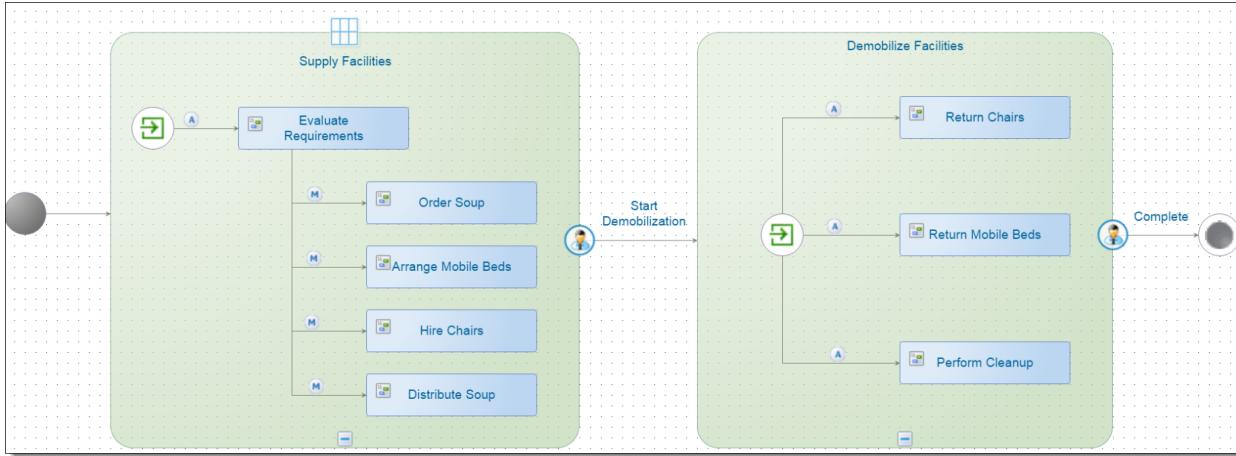
The business logic for this situation is defined based on parameters called Temperature and Duration of the delay. Typically this information is captured during the creation of an item in Process Experience and the data is used in the Lifecycle model. Based on this information, the evaluation of applicability for the follow-up activities is performed.

The following table explains the parameter values and the applicable activities:

		Situation A	Situation B	Situation C
Conditions	Temperature (deg Celsius)	< 5		
	Delay (minutes)	> 120	90-120	<=90
Activities	Order Soup	Recommended	NotRecommended	NotRecommended
	Distribute Soup	Recommended	NotRecommended	NotRecommended
	Arrange Mobile Beds	Recommended	Warning	Warning
	Hire Chairs	NotRecommended	Recommended	Warning

In this table, Situation A states that if the temperature is less than 5 and the delay is more than 120, the recommended activities are Order Soup, Distribute Soup, and Arrange Mobile Beds. The activity Hire Chairs is not recommended.

The following Lifecycle model handles this scenario. It has two states: Supply Facilities and Demobilize Facilities. The Applicability Service is defined, applied to the Supply Facilities state, and implemented using a business process. In the business process, the business logic is implemented using a decision table.



Basic steps to build the Applicability Service

1. Create an entity with the required properties, the Lifecycle build block, and other required building blocks. See [Creating an entity](#).
2. Create the decision table schema fragment. See [Creating the decision table schema fragment](#).
3. Create the Applicability Service using the Applicability contract. See [Creating the Applicability Service using a contract](#).
4. Create a business process with Applicability service as a contract. See [Creating a business process](#)
5. Create the decision table. See [Creating the decision table](#).
6. Update the business process. See [Updating the business process](#).
7. Build the Message Map assignments. See [Building the message map assignments](#).
8. Configure business process as the Applicability Service in the Lifecycle model. See [Building the message map assignments](#).
9. Publish the Lifecycle and open the entity item. See [Publishing the project and executing the entity item](#).

Creating an entity

The first step is to create an entity with the required properties, the Lifecycle build block, and other required building blocks.

To create an entity that has the required building blocks:

1. Create an entity with the properties **Temperature** and **Delay**.
2. Add a Create form that contains the Temperature and Delay properties. This is the form that will be used to create an item in Process Experience.
3. Add a web service building block and select **Read** from the **Basic Operations**.
4. Add the Lifecycle building block and any others that are required by your application.
5. Select the Lifecycle building block, click **Configure Task Entity**, and add the required building blocks.
6. Save the entity.

Creating the decision table schema fragment

The next step is to [create an XML Schema](#) document to define the facilities data with the following XML instance and name it FacilitiesEvaluationSchema.

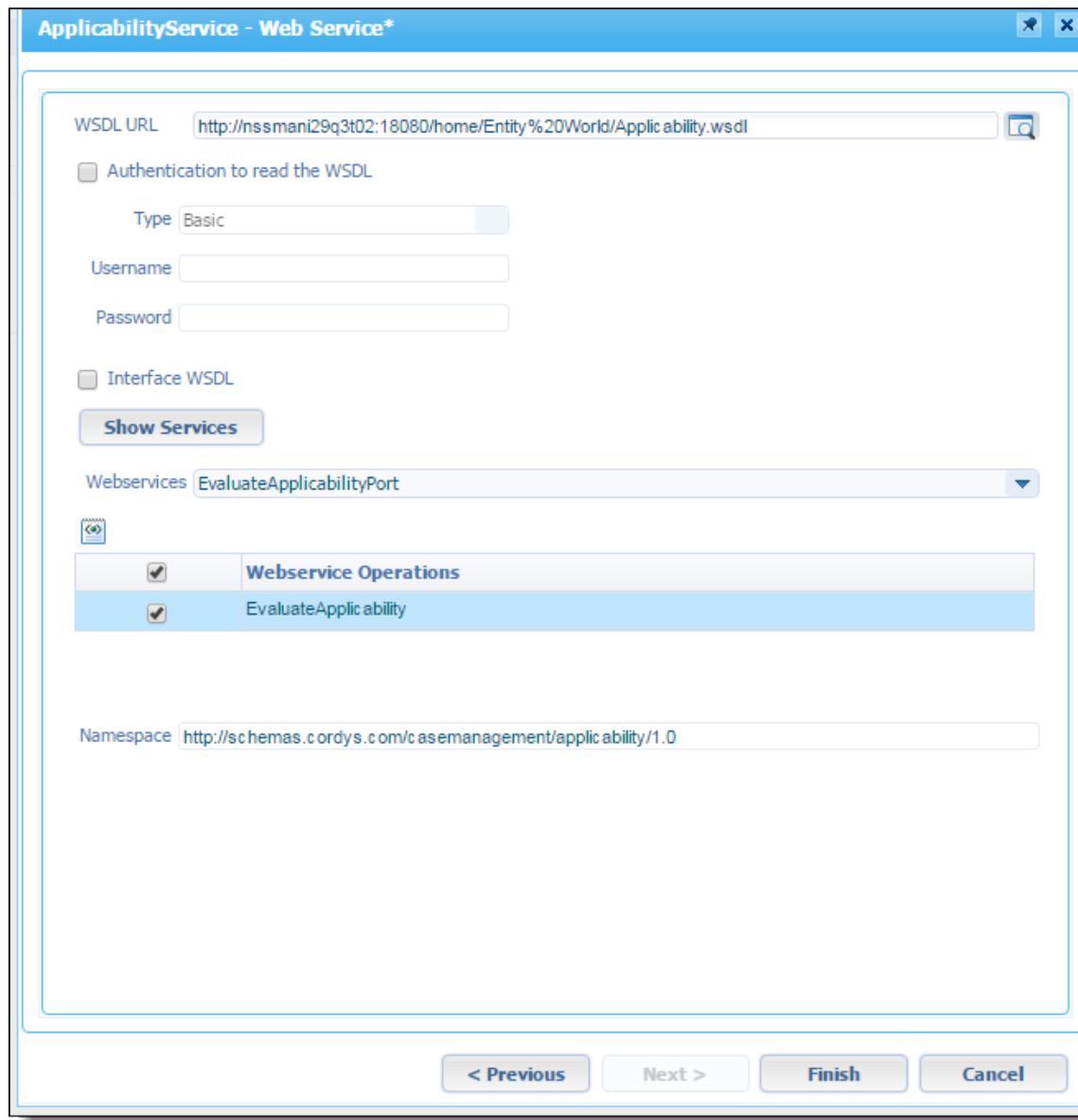
```
<ns:FacilitiesData xmlns:ns="http://schemas.applicability.com/implementation/1.0">
    <ns:EntityData>
        <ns:Temperature>string</ns:Temperature>
        <ns:ExpectedDelay>string</ns:ExpectedDelay>
    </ns:EntityData>
    <ns:Activities>
        <ns:OrderSoup>string</ns:OrderSoup>
        <ns:DistributeSoup>string</ns:DistributeSoup>
        <ns:ArrageMobileBeds>string</ns:ArrageMobileBeds>
        <ns:HireChairs>string</ns:HireChairs>
    </ns:Activities>
</ns:FacilitiesData>
```

Creating the Applicability Service using a contract

The next step is to create the Applicability Service using a contract.

To create the Applicability Service using a contract:

1. Copy the WSDL contract to the development server.
2. Create a new Web service using the above contract.
 - a. Right-click a folder and select **New > Web Service**.
 - b. In Select the source control, select **Import WSDL**, specify the Name and Description, and click **Next**.
 - c. In **WSDL URL**, provide the URL of the WSDL that you copied to the development server, select the web service operation, and click **Finish**.



Creating a business process

Now you need to create a business process with the Applicability service as a contract.

To create a business process with the Applicability service as a contract:

1. Right-click a folder and select **New - Business process model**.
2. Right-click the editor of the business process and select **Properties**.
3. Select the **Contract** check box and click to select the web service that you created.
4. Save the business process.

This generates a skeleton of the business process with the defined contract. You need to implement this logic in the business process. For this example, decision logic will be written.

Creating the decision table

You now need to create the decision table.

To create the decision table:

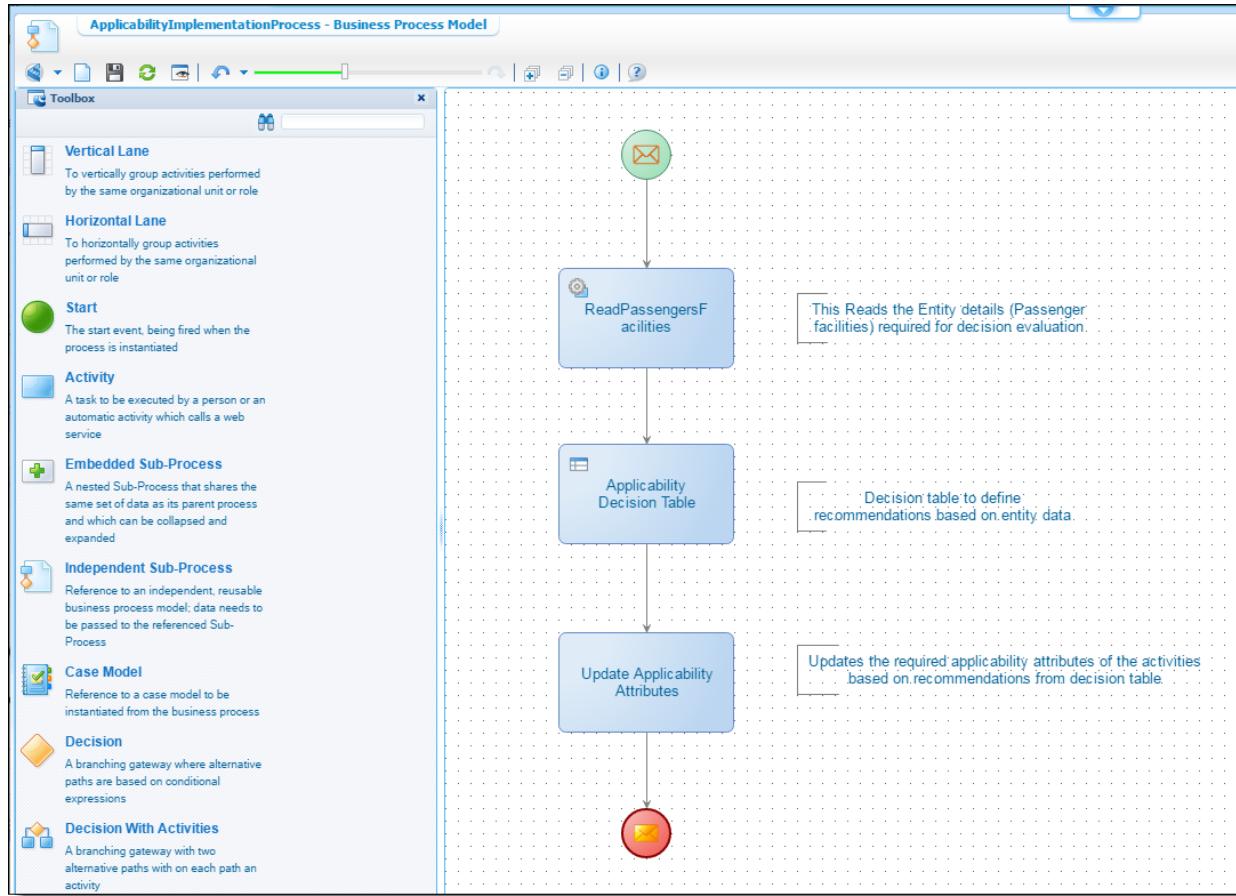
1. Right-click a folder and select **New > Rule Group**.
2. Provide a name and description and save the Rule Group.
3. Right-click the Rule Group and select **Add > Decision Table**.
4. On the left-hand pane of the Decision Table editor, click to open the FacilitiesData schema fragment that was created (see [Creating the decision table schema fragment](#)). The schema fragment is available under the FacilitiesEvaluationSchema XML Schema document.
5. Model the decision table as shown in the following illustration:

Updating the business process

You now need to update the business process that you created.

To update the business process:

1. Access the business process that you created (see [Creating a business process](#)) and insert Web-service generated on the entity (see [Creating an entity](#)) in an empty activity. This replaces the empty activity with the web service activity.
2. Insert the decision table that you generated (see [Creating the decision table](#)) in an empty activity. This replaces the empty activity with the decision table activity.
3. Add a new empty activity as shown in the following illustration.

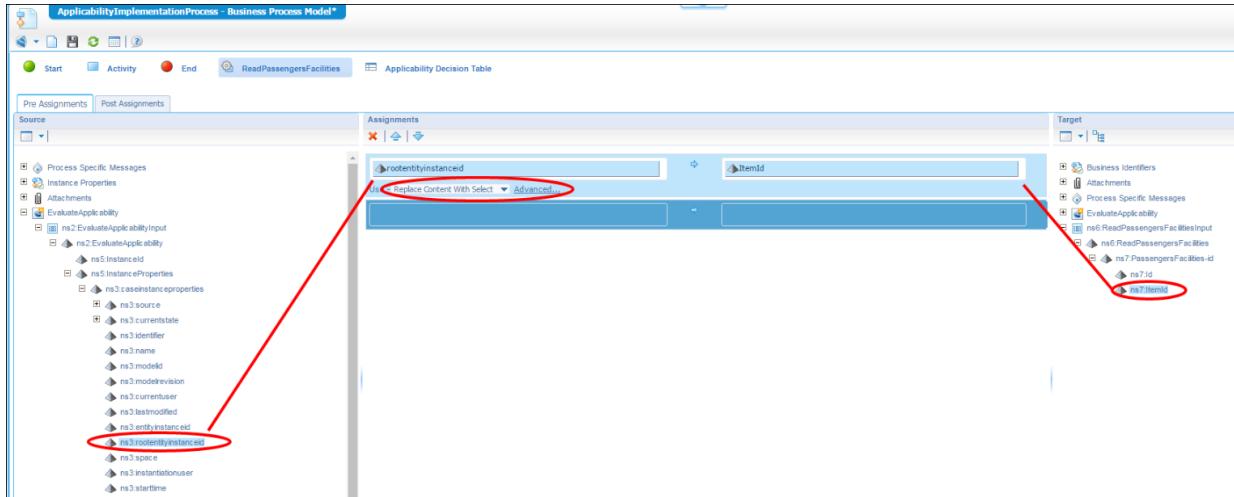


Building the message map assignments

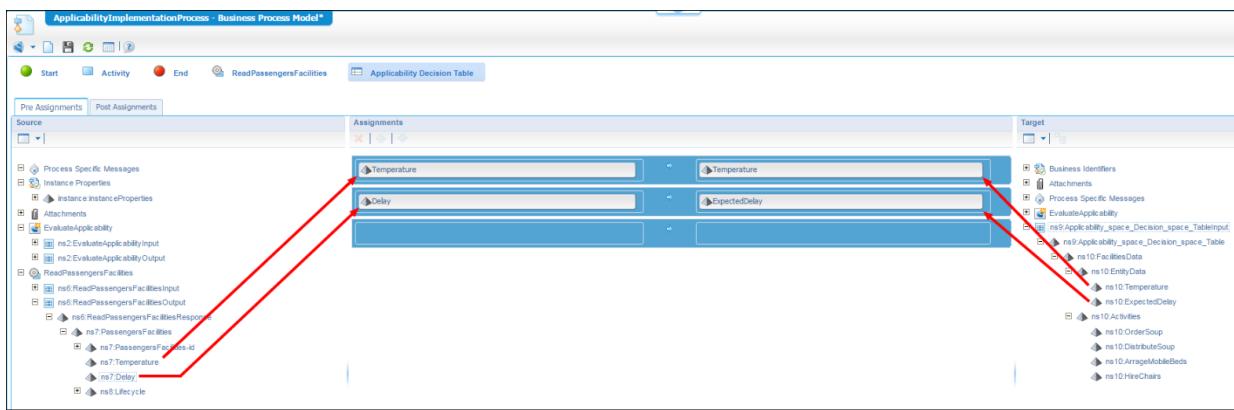
The next step is to build the message map assignments.

To build the message map assignments:

1. Select the **ReadPassengersFacilities** Web-service activity by clicking the activity and go to the Message Map tab.
2. Set the rootentityinstanceid element as the source for the target itemId element of the ReadPassengersFacilitiesInput message as shown in the following illustration.

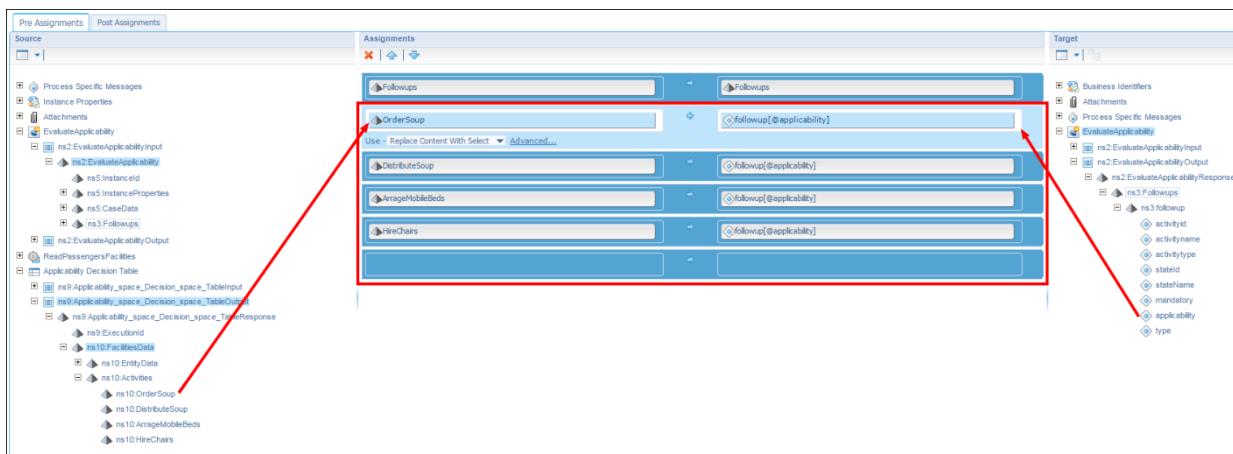
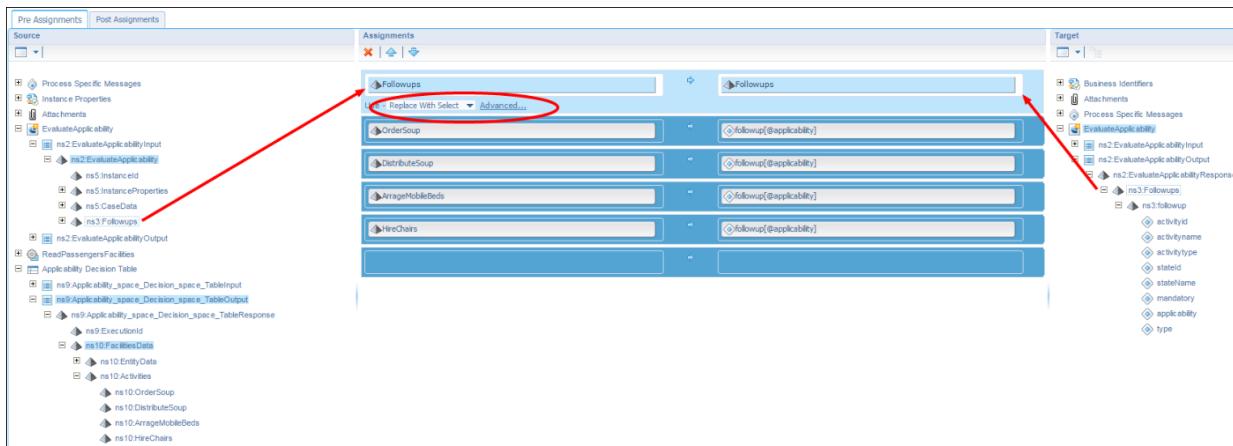


3. Click the **Applicability Decision Table** activity in the activities artifact viewer.
4. Set the **Temperature** element of ReadPassengersFacilitiesOutput as the source for the target Temperature element of the Applicability_space_Decision_space_TableInput Message as shown in the following illustration.
5. Set the **Delay** element of ReadPassengersFacilitiesOutput as source for the target ExpectedDelay element of the Applicability_space_Decision_space_TableInput Message as shown in the following illustration.



6. Click the **Update Applicability Attributes** activity in the activities artifact viewer.
7. Perform the following assignments:
 - a. Replace the **Followups** element of the EvaluateApplicabilityOutput message with the follow-ups of element of the EvaluateApplicabilityInput message.
 - b. Modify the assignment operation to **Replace With Select**.
 - c. Map the activity **Order Soup** element value from Applicability_space_Decision_space_TableOutput message to the applicability attribute of the EvaluateApplicabilityOutput message.
 - d. Modify the assignment operation to **Replace Content with Select**.

- e. Repeat step b for all activity elements, as shown below.
- f. Click **Switch to consolidated view** on the toolbar to switch to the consolidated view.
- g. Update all the target assignments in the target column with expressions as shown in the following illustration.
- h. Save the business process.



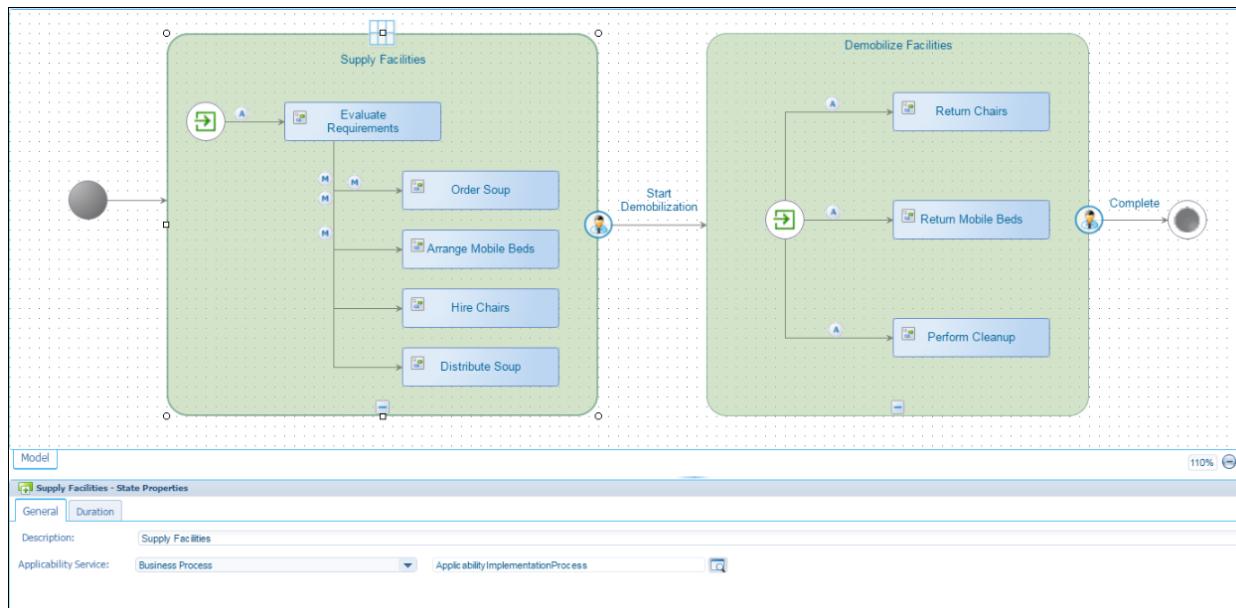
	Target	Use	Source
Start			
Update Applicability Attributes			
ns2:EvaluateApplicabilityOutput/ns2:EvaluateApplicabilityResponse/ns3:Followups	Replace With Select	ns2:EvaluateApplicabilityInput/ns2:EvaluateApplicability/ns3:Followups	
ns2:EvaluateApplicabilityOutput/ns2:EvaluateApplicabilityResponse/ns3:Followups/ns3:followup[@activityname='Order Soup']/@applicability	Replace Content With Select	ns9:Applicability_space_Decision_space_TableOutput/ns9:Applicability_space_Decision_space_TableResponses/ns10:FacilitiesDatas/ns10:Activities/ns10:OrderSoup/text()	
ns2:EvaluateApplicabilityOutput/ns2:EvaluateApplicabilityResponse/ns3:Followups/ns3:followup[@activityname='Arrange Mobile Beds']/@applicability	Replace Content With Select	ns9:Applicability_space_Decision_space_TableOutput/ns9:Applicability_space_Decision_space_TableResponses/ns10:FacilitiesDatas/ns10:Activities/ns10:DistributeSoup/text()	
ns2:EvaluateApplicabilityOutput/ns2:EvaluateApplicabilityResponse/ns3:Followups/ns3:followup[@activityname='Hire Chairs']/@applicability	Replace Content With Select	ns9:Applicability_space_Decision_space_TableOutput/ns9:Applicability_space_Decision_space_TableResponses/ns10:FacilitiesDatas/ns10:Activities/ns10:ArrangeMobileBeds/text()	
ns2:EvaluateApplicabilityOutput/ns2:EvaluateApplicabilityResponse/ns3:Followups/ns3:followup[@activityname='Distribute Soup']/@applicability	Replace Content With Select	ns9:Applicability_space_Decision_space_TableOutput/ns9:Applicability_space_Decision_space_TableResponses/ns10:FacilitiesDatas/ns10:Activities/ns10:HireChairs/text()	
End			
ReadPassengersFacilities			
ns6:ReadPassengersFacilitiesInput/ns6:ReadPassengersFacilities/ns7:PassengerFacilities-ids/m:Item	Replace Content With Select	ns2:EvaluateApplicabilityInput/ns2:EvaluateApplicability/ns5:InstanceProperties/ns3:caseinstaneProperties/ns3:rootentityinstanc eid/text()	
Applicability Decision Table			
ns9:Applicability_space_Decision_space_TableInput/ns9:Applicability_space_Decision_space_Tablens10:FacilitiesDatas/ns10:EntityDatas/ns10:Temperature	Replace Content With Select	ns6:ReadPassengersFacilitiesOutput/ns6:ReadPassengersFacilitiesResponse/ns7:PassengerFacilities/m7:Temperature/text()	
ns9:Applicability_space_Decision_space_TableInput/ns9:Applicability_space_Decision_space_Tablens10:FacilitiesDatas/ns10:EntityDatas/ns10:ExpectedDelay	Replace Content With Select	ns6:ReadPassengersFacilitiesOutput/ns6:ReadPassengersFacilitiesResponse/ns7:PassengerFacilities/m7:Delay/text()	

Configuring the business process as an Applicability Service

The next step is to configure the business process as an Applicability Service in the Lifecycle model.

To configure the business process as an Applicability Service in the Lifecycle model:

1. Create a [Lifecycle model](#).
2. Right-click the **Supply Facilities** state and select **Properties** to go to its properties view.
3. Select the type of Applicability Service that you want to configure (in this case, **Business Process**).
4. Click and select the related business processes (see [Creating the decision table](#)).
5. Save the Lifecycle model.



Publishing the project and executing the entity item

Finally, you need to publish the project and open the item in Process Experience.

To publish the project and execute the entity item:

1. Start the Rule repository Service group.
2. Right-click the project and select **Publish to organization**.
3. Attach the web service interface you created (see [Creating the Applicability Service using a contract](#)) to the BPM Engine Service group.

Note: If you are working in a Non-System organization, ensure that you create a new Business process management Service Group in that organization and attach the Web service interface.

4. Trigger the entity instance from Process Experience with valid parameters from the form.
5. Go to worklist **Facilities List** created for this entity.
6. Open the item that was created.

7. From the **Tasks** panel, try to complete the **Evaluate Requirements** activity.

This invokes the Applicability Service defined on the state and evaluates the provided input. When evaluated, the output information can display the Follow-Ups view with appropriate details as shown in the following illustration.

The screenshot shows a dialog box titled "Add follow up tasks". On the left, under "Available tasks", there is a list of four items: "Order Soup", "Arrange Mobile Beds", "Hire Chairs", and "Distribute Soup". Each item has a small icon and a thumbs-up or thumbs-down icon next to it. On the right, there are several configuration fields: "Assignment type" set to "Individual", "Assigned to" set to "cordys", "Due on" section with "Duration" selected, and input fields for Days (0), Hrs (0), and Mins (0). A "Set default values" button is also present. At the bottom, there are "OK" and "Cancel" buttons.

Adding an assignee

The Assignee building block enables you to specify who is responsible for an item (entity instance). An item can be assigned to a workgroup such as Role and Organization Unit or to

a specific user. It specifies the default assignee, whether e-mail notifications are sent to assignees, and the e-mail template to use for notifications. Each entity can contain only a single Assignee building block.

Important:

- Before you can configure e-mail notifications, you must create the e-mail templates that will be used. See [Creating an email template](#).
- If the Assignee building block is added to an entity, a default assignee must be specified or publishing will fail.

Including the Assignee building block makes the following actions available for an action bar. You can also configure permissions on these actions in the Security Editor. See [Configuring security](#).

Assign - Assigns responsibility for an item that is currently unassigned. If assignee is set to workgroup (Role or Organization unit), the Assign action enables a user (with Assign permission) to assign an item to a user from that workgroup.

Claim - Accepts responsibility for an item that is currently unassigned or assigned to one of the participant's workgroups, setting the assignee to the participant specifically. This action requires Claim permission.

Revoke - Revokes a claimed item. Once revoked, the item moves back to the corresponding workgroup. Only claimed items can be revoked. This action requires Revoke permission.

If the entity also contains a Discussion building block, a Process Experience user is prompted for comments on the revoke action and the comments are captured in the Discussion notes. If the entity does not contain a Discussion building block, the user is not prompted for notes.

Forward - Assigns responsibility for an item that is currently assigned to either the participant individually or to one of the participant's workgroups to another specific participant or workgroup. This action displays a form prompting for the workgroup or individual to be set as the assignee. This action requires Forward permission.

When an item is forwarded, assignee behavior works as follows in Process Experience:

- If an item is assigned to an organization unit, only the user with the lead position in the organization unit can forward the item to a user in the organization unit or to other groups.
- If an item is assigned to a user of an organization unit, the assignee can forward the item to other users in the organization unit.
- If an item is assigned to role or a role user, all the users of the role can forward the item to other users. groups, or group users.
- If item is assigned to a specific user, only that user can forward it to other specific users.

You can create a filtered list based on the filter option provided through List building block. For example,

- To list all the items assigned to the current user, on the Properties tab select the Description property from the AssigneeIdentity relationship. On the Filters tab, use `$(user.Properties.FullName)`
- To list all the entities assigned to the user's role, on the Properties tab select the Name property from the GroupIdentity relationship. On the Filters tab, use `$(targetRoleNames())`
- To list all the entities assigned to the user's organization unit, on the Properties tab select the Name property from the GroupIdentity relationship. On the Filters tab, use `$(targetTeamNames())`.

To add an Assignee building block:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Assignee**.
3. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

4. Click **Configure**.
5. In **Set Default Assignment**, select the **Assignment type** (Organizational unit, Role, User, or Read from property) that is responsible for items.

Note: You can specify using a property to set the assignment type when a user creates an item in Process Experience. Accepted values for this property are Role, User, and Organizational unit. If a user does not specify any of these an error message is displayed.

Organizational units, Roles, and Users are set up in Open Text Directory Services (OTDS). In Process Experience, users can select a different assignee. The Assignee can be Static or Dynamic (Read from property). The available options depend on the selected Assignment type.

- **Static** enables selecting the role while configuring the building block.
- **Read from property** enables selecting a property (such as user ID) whose value is used to set the assignee of an item in Process Experience. This enables a user to choose the assignee dynamically based on a property of the item at run time.

You can set the default assignment as shown in the following examples.

Assignment type	Static/Dynamic	Description
Team	Read from property	The assignee is read from an entity property, for example <code>item.Properties.<<Property Name>></code> . The item property contains the name of the team.
Role	Static	The assignee is chosen from the Select a Role

Assignment type	Static/Dynamic	Description
		dialog box.
	Read from property	The assignee is read from an entity property, for example item.Properties.<<Property Name>>. The item property contains the name of the role.
User	Read from property	The assignee is read from an entity property, for example item.Properties.<<Property Name>>. The item property contains the user name.
Read from property	Read from property	The type of assignee is read from the specified entity property, for example item.Properties.<<Property Name>>.
	Read from property	The assignee is read from the specified entity property, for example item.Properties.<<Property Name>>.

6. In **Notify assignee through email**, specify who will be notified when the item is assigned, or when other actions occur, and select the email template to use for each type of notification.
 - **New assignee** sends a notification to the new assignee when the assignee is set to an individual user. This is not applicable for a claim action.
 - **Former assignee** sends a notification to the former assignee when the assignee is changed.
7. Click  (Save) and close the window.

Activity flow

Using the Activity flow building block, you can define business processes based on a grid structure. Activity flows are intended for business users to manage the processes on their own without the need for IT teams.

An activity flow is an alternate representation of a business process with the following additional features:

- The representation is based on a grid structure with each row corresponding to a step in the business process.
- Understanding and working with activity flows does not require users to have knowledge about BPMN, case management, and so forth.

Adding an Activity flow building block

Use the Activity flow building block to define the sequencing of activities in the context of an entity. By creating an activity flow model, you depict the steps that are needed to achieve a business objective, such as employee onboarding or invoice approval.

When you add an Activity flow building block to an entity, a building block called Task list is added and child entities called Activityflow and LifecycleTask are automatically created. Activityflow is an implicit entity, so you do not have access to configure it although you can configure the LifecycleTask child entity using the **Configure task** option on the Task list building block. See [Configuring the task entity](#) for information about additional building blocks that you can add to the LifecycleTask child entity.

Initiate activity flow is a standard action for the Activity flow building block. It gives Process Experience users the ability to trigger activity flows. See the *Process Experience User's Guide* for a description of each available task.

You can restrict access to this action through security policies that are defined using the [Security building block](#). Like any other action buttons, you can handle the visibility of these actions through the [Action bar](#) presentation.

To add an Activity flow building block:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Activity flow**.
3. Click **Add**.
4. In the activity flow **Details** pane, click **Configure**.
The activity flow grid is displayed.
5. Click **+** (Add).
6. Build the grid by filling in the following components.

Component	Description
Task name	Name of the task to be executed by a person. The name will be visible in the Task panel for the business users.
Previous task	To sequence the steps in business process, use this to link the predecessor task. This enables the business process to determine what should be released when a task is completed. If this option is left as None, the activity will be released as soon as the activity flow is initiated.
Assign to	The assignee for the task. The following options are available: None - The task is not assigned. Assign to role - The task is assigned to a role defined in the workspace. Assign to organizational unit - The task is assigned to the specified

Component	Description
	organizational unit. Read from property - The task is assigned based on entity properties. If you select this option, select one of the following assignment types: <ul style="list-style-type: none">■ Assign via role■ Assign via organizational unit■ Assign directly
User interface	A layout that is created on a child task entity for this task.
Due in day(s)	The Due date for the task in days.

7. Click  (Save).

Note: You cannot publish an activity flow that does not have any activities defined.

Automatically initiating activity flows based on conditions

You can configure activity flows to be initiated automatically based on evaluation of certain conditions on an entity. An activity flow can be initiated either when an item is created or when a condition is satisfied.

You can specify a rule condition using the expression editor. See [Using the expression editor](#).

Note: You cannot use system properties in conditions. Conditions that use the user context convert user to USER during validation.

Important: The properties used in a condition must be from the entity that you are configuring.

To specify conditions for automatic activity flow initiation:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Activity flow**.
3. Click **Configure**.
The **Activities and Settings** tab opens.
4. On the **Settings** tab, specify when to automatically initiate the activity flow.
 - To initiate it when a user creates an item, select **When an instance of this entity is created**.
 - To initiate it when the specified condition is satisfied, select **When a condition is satisfied**.
5. On the **Basic** tab, define the conditions for the rule. See [Using the expression editor](#).
6. Click  (Save).

Tracking changes

The Tracking building block adds information to an entity to track when and by whom the item was created and last modified. Each entity can include only a single Tracking building block.

Tracking is different from History in that tracking properties and relationships are searchable, whereas history is not.

Tracking information is managed through two relationships to the User entity (Created By and Last Modified By) and two date/time properties (Created Date and Last Modified Date). You cannot see or edit the relationships and properties, although they may be visible in forms, lists, and rules.

Tracking can also be added to child entities. If it is added to the child, an option is available to capture the child tracking information as part of the parent entity.

If tracking is added to an entity:

- When creating a list, you can add CreatedDate and LastModifiedDate properties and two relationships to the User entity. For example, it might be helpful to show the user who last modified an order and when it was modified. You can also create a list filter based on creation date and last modified date or based on the relationships. For example, to show all the items created by a particular user, select the Description property from the CreatedBy relationship on the Properties tab and use `$(user.Properties.FullName)` on the Filters tab.
- When creating a form, you can add CreatedDate and LastModifiedDate properties and two relationships to the User entity. For example, a form for creating an order can include the date when it was created and who created it.
- When creating a rule, you can access the CreatedDate and LastModifiedDate properties as `item.Tracking.CreatedDate` and `item.Tracking.LastModifiedDate` and relationships as `item.CreatedBy` and `item.LastModifiedBy`.

To add tracking to an entity:

1. In **Workspace Documents**, open the entity.

2. Click **Add > Tracking**.

If you are adding tracking to a child entity, select **Publish tracking information to parent entity** if you want to capture child tracking information as part of the parent entity.

4. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

5. Click  (Save).

Chapter 5

Adding business logic

Business logic building blocks define rules that filter and present information and specify how the entity will integrate with other parts of Process Platform.

Adding rules

Defining rules that add business logic to an entity is an important part of application development. A rule consists of three parts: an event, a condition, and an action. The action is performed automatically when the event occurs and the specific condition is met. A rule's action is different from the actions available on an entity.

You can specify a rule condition using the expression editor. See [Using the expression editor](#).

Note: System properties cannot be used in rules. Rules that use the user context convert user to USER during validation.

Important: The properties used in a rule must be from the entity that you are configuring.

To add rules to an entity:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Rule**.
3. In the **Rule Properties** pane, provide a **Display Name**, **Name**, and **Description**.
The **Display Name** is used as the rule's default label in forms.
4. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

5. Click **Configure**.

Tip: Since you have not yet configured the rule, a message informs you that it is in an invalid state. Proceed to the next step.

6. In the **When** list, select the event that occurs on the entity to trigger the rule:
 - **A property changes** - The rule is triggered when a form is first loaded and any time a property that is referenced in the rule is changed.

Note: The [Identity](#) building block provides a property called ItemStatus that you can

use to prevent a rule from triggering on an item that is not complete.

- **A relationship changes** - The rule is triggered when the specified relationship is changed. This occurs when a relationship is added or removed.
Note: To Child relationships are not supported.
- **A user triggers an action button** - The rule is triggered when a user clicks an action button such as Approve that sets the property "approved" to true. The Action button is hidden unless the expression evaluates to true and can be restricted through security policies that are defined using the Security building block. Like other action buttons, the visibility of these actions can be handled through the Action bars presentation.

If you want to get input from the user before the action is invoked, select the form to be displayed. An action rule can be displayed in two scenarios.

- In the first scenario, only one item is selected in a Results panel or an item is opened in a form. In this case, all of the existing property values and relationships are displayed on the Action form and all the property and relationship changes made by the user in the Action form are made directly to the current item. When a user clicks the button at the bottom that is named after the Action rule, the rule action is executed. If the user chooses to click the "X" button at the top right of the Action form, the property and relationship values are saved but the rule action is not executed.
- In the second scenario, multiple items are selected in the Results panel and a user clicks an action button. Because multiple items are selected, the current property and relationship values are not displayed in the Action form. When a user clicks the button at the bottom named after the action rule, all the property and relationship values that the user changed in the Action form on each selected item in the Results panel are saved. Then the rule action is executed for each selected item. To Many relationships and action buttons are not displayed on the Action form in this scenario.

The **After actions are performed** option provides the ability to automatically navigate to a different page after the action is performed.

- **Stay on current item** remains on the current item.
- **Close current item** closes the current item and navigates to the previous item in the breadcrumb panel. If not other items are in the breadcrumb panel, it navigates to the home page.
- **Go to URL** enables the user to navigate to a specified URL. This specified URL can be any fully qualified URL.
- **An instance is created** - The rule is triggered when a new instance of the entity is created, just before the new item is written to the database.
- **An instance is initialized** - The rule is triggered when a new instance of the entity is initialized. This is the appropriate event for rules to populate user defined default values before the Create form is displayed.

- **An instance is deleted** - The rule is triggered when an object instance is about to be deleted.

Note: Events are not evaluated on the client side and therefore do not apply to Show error, Show warning, Disable, and Hide rule actions. On the Set and Start Process rule, actions are available when the Action type is selected.

7. On the Basic tab, define the conditions for the rule. See [Using the expression editor](#).
8. In the **Then** area, select an action to perform when the conditions of the rule are met: Show error, Show warning, Show Information, Set, Hide, Disable, Start Process, or Apply Styles. See [Rule actions](#).
 - For Show error, Show warning, or Show information enter a message. The Show information option is available only if **A property changes** is selected in the When list.
 - For Set, enter the target and source.
 - For Disable or Hide, select a property or category.
 - For Start Process, select a process.
 - For Apply Styles, select a property or category.
 - Choose a style (Bold, Italic, Underline) and color for the text.
 - Choose a style (Bold, Italic, Underline) and color for the label.
 - Choose a color for the background.
 - Choose a color for the border.

A preview is shown in the **Preview** area. This option is available only if **A property changes** is selected in the **When** list. The styles that you specify will be shown in Process Experience (in forms and lists). See [Apply styles](#) for more information.

9. Click  (Save).

Rule actions

When you configure a rule, you specify the actions that should occur when the conditions of the rule are met. You also specify the message that should be displayed. Some actions, such as Show warning, make sense only on the client. Others, such as an error that should preclude saving an object instance, might apply equally well on the server or the client. And some might only apply on the server. The following sections describe each rule action in detail.

Show warning

When the Show warning action is triggered, it reports that the entity instance is in a state that requires attention. A Show warning error does not block the saving of any changes to the item; warnings are purely a user interaction enhancement. Rules with this action are run only on the client.

When a user opens an item, all of the Show warning rules are evaluated and any warnings are reported.

The Show warning's action's parameter is the message to be displayed to the user.

Show error

When the Show error action is triggered, it reports that the entity instance is in an error state and requires attention. An error does not block the saving of any changes but can be used to enable or disable rule action buttons. See [Configuring a rule that defines an action button](#). Rules with this action are run on both the client and the server.

On the client, the Show error rule runs as the user manipulates the item and the error is reported immediately when the change is made. When the expression no longer evaluates to true the error message is automatically removed. The user cannot remove the error message.

Show error rules are applied after each property's intrinsic validation. For example, no Show error rule is required to specify that a user cannot type text into an integer property or enter the date of 2/29/2011 into a date property. Further, some property types include options such as minimum and maximum values that are enforced as intrinsic validations.

Show information

When the Show information rule action is triggered, the information message is shown as a modal dialog box in the center of the screen. When the information message is shown, no actions can be performed until the user closes it by clicking **OK**.

An information message is shown when the item is loaded if the information rules evaluated to true or when the property in the condition changed and the condition evaluated to true. If more than one information rule condition is true, the messages are shown in a collapsible list.

Set

A Set rule sets the value of a property to the result of an expression if a condition is true: C = A + B if X = 3. A common special case is when the rule has no condition. In this case, the expression is always true and the rule can be thought of as an assertion of fact. For example, the rule C = A + B states that C is always equal to the sum of A and B. Internally, the rule is triggered on changes to any of the properties referenced by the expression that calculates the new value. In the first example above, the rule must listen for changes to A, B, C, and X.

The Target value is required to be a fully qualified property name for a property defined in the entity. An example target value is: item.Properties.TestIntPropC. The source value can be any valid expression that can be expressed in the expression language. Following is a simple example that multiplies two integer values and stores the result into TestIntPropC:

```
item.Properties.TestIntPropA * item.Properties.TestIntPropB
```

The case of the property name in the expression, source, or target must match the property definition. If you type an incorrect property name or expression, you get an error message describing the error below the input box.

Set rules execute on both the client and the server. They must run on the server to ensure that the object instance is in a correct state when manipulated via APIs. Set rules are also run in our client software to provide responsive feedback, even though they will be executed again when the changes are persisted to the server.

In forms, the target property of a set rule is automatically disabled because its value is being set by the rule and cannot be changed by a user. If the set rule has a condition, the target property is only disabled if the condition is true. A set rule acts as a form disable rule across all of the entity's forms. Consider the rule: $C = A + B$ if $X = 3$, if $X = 3$, then C will be disabled, otherwise it will not be disabled by this rule (it may be disabled by some other rule).

Circular references between rules are not permitted. For example, an object may not simultaneously include the rule $C = A + B$ and $A = C + 3$. All circular references must be resolved before the solution can be used.

Consider the following example:

```
Set item.Properties.Total = item.Properties.A + item.Properties.B
```

This rule references the properties A and B in the entity. The system triggers this rule automatically whenever the value of A or B is changed. When the rule is triggered the value of Total is recalculated.

It is possible to have multiple rules with the same target property because those rules will most likely have matched conditions. For example:

Rule 1: $C=A+B$ with condition $D==true$

Rule 2: $C=A$ with condition $D==false$

In this example both rules do not evaluate to true at the same time.

It is possible to create rules, with the same target, having unmatched conditions. These may evaluate to true at the same time and produce unexpected results. This cannot be verified at design time.

When the target is an enumerated value, the source value must be the internal value instead of the display name. The display name may change depending on the locale so the internal value is required.

Hide

In Process Experience, hides the selected property or the form components tagged with a specific category when the rule evaluates to true.

Disable

In Process Experience, disables the selected property or the form components tagged with a specific category when the rule evaluates to true.

Start Process

Important: Effective with Release 16.3, business processes are closed by default. Therefore, each business process called from a rule needs to have the required permissions. To configure security, right-click the business process and select **Define Runtime Security**.

A Start Process rule action can be used to initiate a Process Platform Business Process given a condition, event, or user action. When the Start Process action is triggered the Business Process is executed on the server.

The Start process rule action requires the user to select the Business Process by clicking a Find button to open the Select Process dialog box. Either select an existing Business Process in the dialog box or click **New** to create a new Business Process. See [Integrating BPM processes and entities](#) for additional details.

Options are available that enable you to specify when and how often the Start process rule action is evaluated, and possibly triggered, when the Rule event is configured as **on property changes**.

- The **edge-trigger** functionality enables you to configure the Start process rule to be triggered only when the result of the expression changes from false to true. The other option is to configure the Start process rule to be triggered when one of the properties referred to in the expression is modified and the result of the expression is true.
- The **trigger a Start process rule action immediately** functionality enables you to configure the Start process rule to be triggered immediately when a property is modified on a form. The default behavior for the Start process rule action is to only evaluate, and possibly trigger, the rule expression when the instance is saved to the server. This usually occurs on a form when the user moves the pointing device outside the form area.

The Start process rule action cannot be saved to the server when the item is in an error state. When the errors are resolved, the item is saved to the server and the Start process rule action rule condition is evaluated and possibly triggered.

When the process is successfully started, a process ID is returned. Optionally, the Start Process rule action can store this process ID in a text property. The Start Process ID option enables you to save the ID of the process instance that is created by the rule's action as the value of a property in the entity itself. The list presents all of the entity properties that have the right data type (Text/String) to store the process instance ID.

This setting is optional. It can be used for two purposes:

- It provides access to the process instance (so that it can be manipulated by other processes that may also be running against the entity instance).
- It provides a transactional safe way to know that a process has been triggered. You can use a "property not null" condition on the action to prevent the process from being fired more than once (important if it is a user action where the user clicks the button to start the process).

Start Process rule actions execute on the server by making an ExecuteProcess Soap API call passing the identifiers of the current item as part as the payload. This information can be stored in the Process instance by creating a message map. Once stored in the message map the ID can be used during the execution of the Business Process to make web service calls to get and update the item instance property values.

An example of the payload follows:

```
<Invoice-id xmlns:ns="http://schemas.cordys.com/bpm/execution/1.0"
             xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
             xmlns="http://schemas.cordys.com/default">
  <Id>16388</Id>
  <ItemId>005056C0000811E5E4962EA28B7F9689.16388</ItemId> <
  EntityType>Invoice</
  EntityType > </Invoice-id>
```

Note: The Start message payload requires the default xml namespace of `http://schemas.cordys.com/default`. The Start message XML schema in the Business Process must match this namespace.

To add the message map:

1. Open the business process model.
2. Select the **Message Map** tab.
3. Expand **Process Specific Messages**.
4. Right-click the Process Specific Messages entry and then click **Create Message**. Name the new message "`<-entity-name->-id`" where `<-entity-name->` is replaced by the actual name of the entity that is starting the process. When saved, the message will be prefixed with the bpm namespace. The name is case sensitive.
5. Right-click the "`<-entity-name->-id`" entry and CreateElement. Name the new element "Id". When saved, the element will be prefixed with the bpm namespace.

Optionally, you can add the ItemId and Level elements.

Apply styles

In Process Experience, a property can be displayed with formatting that you specify. You select the property or category and then choose the style and color for the text and label and a color for the background and border when configuring the rule. Styles applied on properties are respected in both forms and lists, but styles applied on categories are applicable only to forms.

For example, formatting can be used in a Process Experience form to emphasize a property where user input is required or to highlight data entry that is not within a certain value range. You can apply styles to a property or category. When you apply styles to a category, they are applied only to property controls in forms (such as Input, Date, Radio button, and so forth) within the specified category.

If you apply styles to a property or category in both a rule and a form, the formatting in the rule overrides the formatting in the form. For example, if you configure a form to format a property called Amount with a green background and a rule that configures Amount to have a purple background, the color configured in the rule (purple) overrides the color configured in the form (green). Therefore, the property has a purple background when a user opens a form that contains it.

However, assume that a rule specifies that the Amount property should have a purple background only if the value of a property called Quantity is between 1 and 10. When the rule runs, if the value of Quantity is 5, Amount is shown with a purple background (not the green background specified in the form). If the value of Quantity changes to 12, when a user opens the form and the rule runs, the property reverts to the green background specified in the form.

Configuring a rule that defines an action button

Configuring an action rule that sets a property or business process model makes an action button available for adding to an action bar. For example, if a vendor entity has an enumerated text property with values of None, Active, and Inactive you might create a rule that enables a Process Experience user to change a vendor status from None to Active. For this type of rule, the Display Name becomes the caption on the action button (in this case, Activate).

When you save the rule, an option called Activate is available for adding to an action bar that is included in an Actions panel for a layout. When the rule evaluates to True (the vendor status is None), an Activate button is then available on the action bar specified in the Actions panel for the layout.

The following procedure uses this example.

To configure a rule that defines an action button:

1. Create the rule and give it a **Display Name** that you want to be the caption for the action button.
2. In the list of building blocks, select the rule and click **Configure**.
3. In the When list, select **A user triggers an action button**.
4. Select **Allow this action when there are errors** to allow the action button to be enabled even when the current item is set into an error state by a defined rule.
An action button is disabled regardless of this option if a property value fails intrinsic validation. An example of intrinsic validation failing is when a user types text into an integer property. By default, all rule action buttons are disabled when a rule error sets an item into the error state.
5. If you want to display a form, select it in the **Display form** list. This is optional.
6. In the **If** area, select **Status > equal to > None**.
7. In the **Then** list, select **Set**.
8. In **Target**, type **Status**. The name is automatically changed to item.Properties.Status (or you can type it this way directly).

9. In **Source**, type "**Active**" - be sure to include the quotation marks.
10. Save the rule.

Using system level properties in rules

System level properties are available for use in applications. Typically, the value of these properties depends on the environment where the application is installed. The following system properties are available:

- system.baseURL is the beginning URL portion for references to other OpenText applications running on the same server in the same organization.
- system.organization is a reference to the current user's organization.

Using the User entity properties in rules

The User entity has several properties and relationships that provide information about the current user. These properties can be used in rule expressions. Some common examples are:

- User.Properties.UserId
- User.Properties.FullName
- User.toPerson.Properties.Email

Using durations to specify relative dates and times

Durations provide the ability to specify relative dates and times. For example, a library book might be due two weeks after the book is checked out or a step in a project might have an expected completion date of the actual start date plus the expected duration, where the expected duration was entered by the user and the start date was when the form was completed. These examples can all be accomplished using Duration properties and constants in rules and expressions.

Durations can be ambiguous without an anchoring date. For example, "1 month" could be 28-31 days depending on which date you add it to. Therefore, Durations cannot be compared to one another, or anything else, in an expression. This is also why duration properties are not searchable in a list query.

The following examples show how to use Duration properties in rules and expressions. These examples assume that an entity contains StartDate and EndDate date properties and a Duration property, and that all of these properties appear on a form.

The following rule takes the StartDate property and adds the Duration property to determine the End Date property:

```
Set item.Properties.EndDate = item.Properties.StartDate + item.Properties.Duration
```

The following Show warning rule cautions the user that the combination of the StartDate and the Duration does not fall within the calendar year:

Show warning "Past the end of the year"

If item.Properties.StartDate + item.Properties.Duration > item.Properties.StartDate +
1 end of years

Given the previous Set rule, you might assume that the condition of the Show warning rule could use item.Properties.EndDate in place of item.Properties.StartDate + item.Properties.Duration, since they are now the same value, resulting in:

If item.Properties.EndDate > item.Properties.StartDate + 1 end of years

However, when the warning appears, the first example would highlight the StartDate and Duration fields on the form, whereas the second example would highlight the StartDate and the EndDate fields on the form. And the EndDate field is not modifiable since it is the target of a set rule. So the user might not know that the warning can be satisfied by reducing the duration, which would change the offending EndDate. Both of the examples also show how one or more durations can be compared to each other indirectly if there is some date to give them specific meaning, since both of the above effectively state "If Duration is greater than 1 end of years" which would not be allowed.

While absolute dates are of limited utility, constant values for durations are quite useful. Similar to above, where the end date had to be in this calendar year, perhaps another requirement is that there be at least 10 days allowed. This is an example of adding a duration constant to the StartDate.

In this example, the sum of the StartDate and a specific duration of "10 days" is used in the comparison.

The duration constant constructor takes either a single [ISO-8601 Duration](#) specification as a quoted string or 1-6 integers, each integer representing one of the following units: Years[, Months[, Days[, Hours[, Minutes[, Seconds]]]]]. For example, three integers represent Years, Months, and Days.

An example of specifying a duration in ISO format follows:

Show warning "Too soon!"

If item.Properties.EndDate < item.Properties.StartDate + duration("P10D")

The same can be expressed using the three-argument version of the duration constant constructor with 1 year, 2 months, 3 days, where the omitted hours, minutes, and seconds are all assumed to be 0.

Show warning "Too soon!"

If item.Properties.EndDate < item.Properties.StartDate + duration(1, 2, 3)

This example is, once again, an indirect comparison of the independently invalid "If Duration is less than 10 days".

Using To Many relationships in rules

Important: Some programming knowledge is required to include the information described in this topic in your application.

To Many relationships are tricky to use in rules because they represent a collection of items rather than a single item.

Although you can iterate using a process, the rules engine addresses some use cases by providing operations that work on collections as a whole. You get a collection using empty brackets:

```
item.<-relation->[].Properties.<-property->
```

This returns an array of property values. The rules engine provides array and aggregation operators to work with these collections. For example, to get the total amount of an invoice:

```
sum(item.Line[].Properties.Amount)
```

See [Arrays and aggregation](#) for a list of available operators.

Using advanced functions

The following functions can be called from rules. However, they cannot be used in On Property Change or in a Set Rule in the [Rule](#) building block. The functions can also be used in the [List](#) building block (enclosed in \$()) to specify a filter. In Process Experience, the filter form does not provide a way to enter advanced functions to narrow down the list within your application.

Note: These functions can be used only on server side rules, which effectively means they can only be used in a condition expression in a security policy.

Function	Description
targetUser()	Returns the ID of the current runtime user.
targetRoles()	Returns the roles of the current runtime user.
targetTeams()	Returns the teams of the current user.
targetWorkLists()	Returns the worklists of the current user.
targetAll	Returns the current user and the user's roles, worklists, and teams.
getCurrentUserLocale()	Returns the user's current locale.
typeOf(entity-id)	Given an entity ID, returns the entity's name.
isInRole(package-name, role-name)	Given a package name and a role name, returns true if the user is a member of the role, false otherwise.
isInGroup(group-name)	Returns true if the current user is a member of the group, false otherwise.

Function	Description
isInOrganizationalUnit(unit-name)	Returns true if the current user is a member of the organizational unit, false otherwise.
GetEntityTypeIds(entity-name, solution-name)	Returns the entity ID given its name and its solution name.
GetEntityTypeIds(list{comma-separated-list-of-entity-names}, solution-name)	Returns a list of entity IDs given a list of entity names and the solution name.
GetEntityTypeNames(entity-id, solution-name)	Returns the entity's name given its ID and solution name.

Using web services

Web services provide a standard means for software applications running on a variety of platforms and frameworks to work together. Web services are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions, thanks to the use of XML. Web services can be combined in a loosely coupled way to achieve complex operations. Programs providing simple services can interact with each other to deliver sophisticated added-value services.

Note: Web services on EIS entities are not supported

To integrate your application with other systems or parts of Process Platform you can use web services. For entities, you can also expose some web service operations to enable interaction between the entities and the other systems or parts of Process Platform.

If an entity has a Security building block, you need a role with the right permissions to use the web services.

Basic web service operations

Web service operation	Entity permissions			Property permissions		Relationship permissions	
	Create	View	Delete	Read	Update	Read	Update
Create<entity>	✓			✓	✓	✓	✓
Read<entity>		✓		✓		✓	
Update<entity>		✓		✓	✓	✓	✓
Delete<entity>		✓	✓				

Relationship web service operations

Web service operation	Entity permissions			Property permissions		Relationship permissions	
	Create	View	Delete	Read	Update	Read	Update
ToOne relationship - Set<relationship>		✓				✓	✓
ToOne relationship - Clear<relationship>		✓				✓	✓
ToMany relationship - AddTo<relationship>		✓				✓	✓
ToMany relationship - Get<relationship>		✓		✓		✓	
ToMany relationship - RemoveFrom<relationship>		✓				✓	✓
ToChild relationship - Create<entity>	✓			✓	✓	✓	✓
ToChild relationship - Get<entity>		✓		✓		✓	
ToChild relationship - Delete<entity>		✓	✓	✓		✓	
ToParent relationship - Get<relationship>		✓		✓		✓	

Find web service operations

A Find web service operation can be added only after the Web service building block has been added.

Web service operation	Entity permissions			Web service operation permission
	Create	View	Delete	Use
<Any Find web service operation name>		✓		✓

Important: When you use multi-byte characters in the names of entities, properties, and so forth, you must instruct Process Platform not to generate errors if such characters result

in invalid URIs. You can do this in the Process Platform Management Console by setting (or creating) the following platform property:

```
bus.xml.nom.suppress.invaliduri.error=true
```

See the *Management Console* section in the Process Platform documentation for detailed information.

The following operations can be exposed as a web service on an entity:

- [Read](#) - Reads a single entity
- [Create](#) - Creates a new entity
- [Update](#) - Updates an existing entity
- [Delete](#) - Deletes an existing entity
- [Find](#) - Reads all entities fulfilling specified criteria.

Each of the following operations can be exposed as a web service on a relationship:

- [To One relationships](#) - Set, Clear
- [To Many relationships](#) - AddTo, Get, and RemoveFrom
- [To Child relationships](#) - Create, Get, and Delete
- [To Parent relationships](#) - Get

Note: Web service operations on relationships to EIS entities are not supported.

See [Entity web service details](#) and [Integrating BPM processes and entities](#) for additional information.

To expose a basic or relationship web service operation:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Web Service**.

The namespace of the web service and the available operations are listed in the Web service details pane. The namespace uses the package name, as specified in the package properties, and the entity name (see [Entity web service details](#)):
`http://schemas/<packagename>/<entityname>/operations`

Note: After the web service operations have been used, the package name should not be changed. This would change the namespace and require an update of the places where they are used.

3. Select (check) the operations to be exposed to the entity.
4. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

5. Click  (Save).

Note: Find web services can only be added after the Web service building block is added.

To remove a basic or relationship web service operation:

1. In **Workspace Documents**, open the entity.
2. Select **Web service**.
The operations are listed in the Web service details pane.
3. Clear (uncheck) the operation you want to remove.
If the operation is in use, you are prompted to resolve this before deleting the operation.
4. Click  (Save).

You can delete the entire Web service building block if all operations are cleared. If not all operations are cleared, you are notified to clear them first.

To create and expose a Find web service operation:

1. In the Web service details pane, click **Find**.
A table containing a toolbar and a column heading called Name is displayed.
 2. In the toolbar click  (Add).
A row is added to the table.
 3. Select the added row and then type a name for the new Find web service operation.
- Note:** It is best practice to use a strict convention for the naming of Find web service operations. For example, always prefix the names with Get or Find. This makes it much easier to locate them in the lists of web service operations that are shown in various places in Process Platform.
4. In the toolbar, or in the selected row, click  (Configure).
The Entity Web Service Find Operation dialog box opens.
 5. On the Filter tab, create an expression and click  (Save).
- See [Configuring a Find web service operation](#).

Note: If a property that is used in a Find web service operation is deleted, the expression used for it becomes invalid. This will be detected when you try to validate or publish the entity.

Because you are completely free to define a Find web service operation, it is not guaranteed that it will perform on a database that contains a large number of entity instances. Your database administrator should be alerted if performance deteriorates, so that measures can be taken to restore performance.

To update a Find web service operation:

1. In **Workspace Documents**, open the entity.
2. Select **Web service**.
The Web service details pane opens.
3. Click **Find**.
A table displays the available Find web service operations.
4. In the toolbar, or in the selected row, click  (Configure).
The Entity Web Service Find Operation dialog box opens.

5. On the Filter tab, create an expression and click  (Save).

See [Configuring a Find web service operation](#).

To remove a Find web service operation:

1. In **Workspace Documents**, open the entity.
2. Select **Web service**.
The Web service details pane opens.
3. Click **Find**.
The available Find web service operations are shown.
4. Select the row with the operation you want to remove.
5. In the toolbar or in the selected row, click  (Delete).
If the operation is in use, you are prompted to resolve this before deleting the operation.
6. Click  (Save).

You can delete the entire Web service building block if all operations are cleared. If some operations are not cleared, you are prompted to clear them.

Web services on entities

The following operations can be exposed as a web service on an entity:

- [Read](#) - Reads a single entity
- [Create](#) - Creates a new entity
- [Update](#) - Updates an existing entity
- [Delete](#) - Deletes an existing entity
- [Find](#) - Reads all entities that match specified criteria.

Entity web service Create operation

The Create operation is intended to create a new entity. The name of the operation is composed as `Create<entity name>`.

See [Entity web service details](#) for more information about representing property values in SOAP requests and responses.

SOAP request

The request takes one argument for the relationship, which is either the Id or the ItemId of the related entity (if both are provided, the ItemId will be used).

The values of the properties are passed in the request. Assume that an Order entity has a [To One relationship](#) named Customer to entity BusinessPartner.

Properties in the request can be set only when the **Allow participants to change the value** is set to **Anytime** or **Only when first created**, see [Adding properties](#).

An example request will then be as follows:

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <CreateOrder xmlns="http://schemas/OpenTextOrderManagement/Order/operations">
      <ns0:Order-create xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
        <ns0:Amount>400.0</ns0:Amount>
        <ns0:ShippingDate>2015-09-14Z</ns0:ShippingDate>
        <ns0:Customer>
          <ns1:BusinessPartner-id
            xmlns:ns1="http://schemas/OpenTextOrderManagement/BusinessPartner">
            <ns1:Id>1</ns1:Id>
          </ns1:BusinessPartner-id>
        </ns0:Customer>
      </ns0:Order-create>
    </CreateOrder>
  </SOAP:Body>
</SOAP:Envelope>

```

For each property, a value can be specified. If the property is not mentioned in the request, it gets the default value. If no default value was specified, its value is null.

SOAP response

The response contains the properties of the new entity. All properties are returned independent of the **Allow participants to change the value** setting. It also contains the generated ItemId. An example response follows.

```

<wstxns1:CreateOrderResponse
  xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <wstxns2:Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
  xmlns="http://schemas/OpenTextOrderManagement/Order"
  xmlns:wstxns2="http://schemas/OpenTextOrderManagement/Order">
    <Customer xmlns="http://schemas/OpenTextOrderManagement/Order">
      <wstxns3:BusinessPartner-id
        xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">
        <ns1:Id xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">1</ns1:Id>
        <ns1:ItemId
          xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">00505601024C11E6F0CA5D
        24D3789644.1</ns1:ItemId>
      </wstxns3:BusinessPartner-id>
    </Customer>
    <Order-id xmlns="http://schemas/OpenTextOrderManagement/Order">
      <ns1:Id>1</ns1:Id>
      <ns1:ItemId>00505601024C11E6F0CA5D24D37E3644.1</ns1:ItemId>
    </Order-id>
    <ShippingDate xmlns="http://schemas/OpenTextOrderManagement/Order">2015-09-
    14Z</ShippingDate>
    <Amount xmlns="http://schemas/OpenTextOrderManagement/Order">400.0</Amount>
  </wstxns2:Order>
</wstxns1:CreateOrderResponse>

```

Entity web service Read operation

The read operation is intended to read a single entity. The name of the operation is composed as `Read<entity name>`.

See [Entity web service details](#) for information about representing property values in SOAP requests and responses.

SOAP request

The request takes one argument, which is either the Id or the ItemId of the entity (if both are provided, the ItemId will be used). The following example reads the Order entity with Id = 1:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <ReadOrder xmlns="http://schemas/OpenTextOrderManagement/Order/operations">
      <ns0:Order-id xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
        <ns0:Id>1</ns0:Id>
      </ns0:Order-id>
    </ReadOrder>
  </SOAP:Body>
</SOAP:Envelope>
```

SOAP response

The response contains the properties of the entity. If the entity has To One relationships with other entities, the IDs of these entities are also included in the response. If the entity has To Many or To Child relationships with other entities, the IDs of these entities are not included - they must be retrieved with the `Get<relationship name>` operation.

See the following example:

```
<wstxns1:ReadOrderResponse
  xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <wstxns2:Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
    xmlns="http://schemas/OpenTextOrderManagement/Order"
    xmlns:wstxns2="http://schemas/OpenTextOrderManagement/Order">
    <Customer xmlns="http://schemas/OpenTextOrderManagement/Order">
      <wstxns3:BusinessPartner-id
        xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner"
        xmlns:wstxns3="http://schemas/OpenTextOrderManagement/BusinessPartner">
        <Id xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">1</Id>
        <ItemId
          xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">00505601024C11E6F0CA5D
          24D3789644.1</ItemId>
        </wstxns3:BusinessPartner-id>
      </Customer>
    <Order-id xmlns="http://schemas/OpenTextOrderManagement/Order">
```

```

<Id>1</Id>
<ItemId>00505601024C11E6F0CA5D24D37E3644.1</ItemId>
</Order-id>
<ShippingDate xmlns="http://schemas/OpenTextOrderManagement/Order">2015-09-
14Z</ShippingDate>
<Amount xmlns="http://schemas/OpenTextOrderManagement/Order">400.0</Amount>
</wstxns2:Order>
</wstxns1:ReadOrderResponse>

```

Entity web service Update operation

The update operation is intended to update an existing entity. The name of the operation is composed as `Update<entity name>`.

See [Entity web service details](#) for more information about representing property values in SOAP requests and responses.

SOAP request

The request takes one argument for the relationship, which is either the Id or the ItemId of the related entity (if both are provided, the ItemId will be used). The values of the properties are passed in the request. Properties in the request can only be set when **Allow participants to change the value** is set to **Anytime**. See [Adding properties](#).

An example request follows:

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <UpdateOrder xmlns="http://schemas/OpenTextOrderManagement/Order/operations">
      <ns0:Order-id xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
        <ns0:Id>1</ns0:Id>
      </ns0:Order-id>
      <ns0:Order-update xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
        <ns0:Amount>500.0</ns0:Amount>
        <ns0:ShippingDate>2015-09-14Z</ns0:ShippingDate>
        <ns0:Customer>
          <ns1:BusinessPartner-id
            xmlns:ns1="http://schemas/OpenTextOrderManagement/BusinessPartner">
            <ns1:Id>1</ns1:Id>
          </ns1:BusinessPartner-id>
        </ns0:Customer>
      </ns0:Order-update>
      <old>
        <ns0:Order xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
          <ns0:Order-id>
            <ns0:Id>1</ns0:Id>
          </ns0:Order-id>
          <ns0:Amount>400.0</ns0:Amount>
          <ns0:ShippingDate>2015-09-14Z</ns0:ShippingDate>
          <ns0:Customer>
            <ns1:BusinessPartner-id>

```

```

<ns1:BusinessPartner-id>
    <ns1:Id>1</ns1:Id>
</ns1:BusinessPartner-id>
<ns0:Customer>
    <ns0:Order>
        </old>
    </UpdateOrder>
</SOAP:Body>
</SOAP:Envelope>

```

The `UpdateOrder/Order-update` contains the new values of the property. The entity to be updated is specified by its ID (in the request: `UpdateOrder/Order-id/Id`). Properties of the entity that are not mentioned are left unchanged.

The `UpdateOrder/old` is optional. It contains values that can be used for detecting concurrent update conflicts. Conflict detection is performed only for properties that are mentioned in the request. See [When users work on this property at the same time, report conflicts](#) for more information.

SOAP response

The response contains all the properties of the updated entity, independent of the **Allow participants to change the value** setting. An example response follows:

```

<wstxns1:UpdateOrderResponse
xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <wstxns2:Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
xmlns="http://schemas/OpenTextOrderManagement/Order"
xmlns:wstxns2="http://schemas/OpenTextOrderManagement/Order">
        <Customer xmlns="http://schemas/OpenTextOrderManagement/Order">
            <wstxns3:BusinessPartner-id
xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner"
xmlns:wstxns3="http://schemas/OpenTextOrderManagement/BusinessPartner">
                <Id xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">1</Id>
                <ItemId
xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">00505601024C11E6F0CA5D
24D3789644.1</ItemId>
                </wstxns3:BusinessPartner-id>
            </Customer>
            <Order-id xmlns="http://schemas/OpenTextOrderManagement/Order">
                <Id>1</Id>
                <ItemId>00505601024C11E6F0CA5D24D37E3644.1</ItemId>
            </Order-id>
            <ShippingDate xmlns="http://schemas/OpenTextOrderManagement/Order">2015-09-
14Z</ShippingDate>
            <Amount xmlns="http://schemas/OpenTextOrderManagement/Order">500.0</Amount>
        </wstxns2:Order>
    </wstxns1:UpdateOrderResponse>

```

Entity web service Delete operation

The Delete operation is intended to delete an existing entity. The name of the operation is composed as `Delete<entity name>`.

See [Entity web service details](#) for more information about representing property values in SOAP requests and responses.

SOAP request

The values of the properties are passed in the request. An example request follows:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <DeleteOrder xmlns="http://schemas/OpenTextOrderManagement/Order/operations">
      <ns0:Order xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
        <ns0:Order-id>
          <ns0:Id>1</ns0:Id>
        </ns0:Order-id>
        <ns0:Amount>500.0</ns0:Amount>
        <ns0:ShippingDate>2015-09-14Z</ns0:ShippingDate>
        <ns0:Customer>
          <ns1:BusinessPartner-id
            xmlns:ns1="http://schemas/OpenTextOrderManagement/BusinessPartner">
            <ns1:Id>1</ns1:Id>
          </ns1:BusinessPartner-id>
        </ns0:Customer>
      </ns0:Order>
    </DeleteOrder>
  </SOAP:Body>
</SOAP:Envelope>
```

The entity to be deleted is identified by the ID (in the request: `Order/Order-id/Id`). The property values are optional, and are used for conflict detection. For each property that is specified, the entity can be deleted only if the property has that specified value. See [When users work on this property at the same time, report conflicts](#) for more information.

SOAP response

On successful delete, an empty response is returned as in the following example:

```
<wstxns1:DeleteOrderResponse
  xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
```

If the identified entity does not exist, a SOAP fault is returned.

Entity web service Find operation

A Find operation is intended to get a list of all entity instances that fulfill specific criteria. The name of the operation can be any name.

See [Entity web service details](#) for more information about representing property values in SOAP requests and responses.

SOAP request

If no parameters were used in the configuration of the web service operation, the request takes no arguments. The criteria to find the instances are captured in the expression defined with the expression editor (see [Configuring a Find web service operation](#)).

An example request will then be as follows.

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <GetOrders xmlns="http://schemas/OpenTextOrderManagement/Order/operations">
      </GetOrders>
    </SOAP:Body>
  </SOAP:Envelope>
```

If parameters were used in the configuration of the web service operation, the request will show them as in the following example.

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <GetOrders xmlns="http://schemas/OpenTextOrderManagement/Order/operations">
      <parameter1>PARAMETER</parameter1>
      <parameter2>PARAMETER</parameter2>
    </GetOrders>
  </SOAP:Body>
</SOAP:Envelope>
```

SOAP response

The response contains all entities that fulfill the criteria.

Testing and deploying entity-based web service operations

Although it is possible to successfully create an entity web service, it is not possible to test the web service operations until a service group with a service container of type

Application Server Connector has been created in the organization and the entity web service interface has been added to the service group.

Be sure to assign the service container to the OS process Application Server by selecting the **Assign OS Process** check box.

After these steps have been taken, the entity web service operations can be tested and consumed by other applications.

For details about how to create a service group and how to create and configure a service container, see the Process Platform product documentation.

To access the documentation, click  (Help) on the shortcut bar of the Process Platform interface.

Web services on relationships

The following operations can be exposed as a web service on a relationship:

- [To One relationships](#) - Set, Clear
- [To Many relationships](#) - AddTo, Get, and RemoveFrom
- [To Child relationships](#) - Create, Get, and Delete
- [To Parent relationships](#) - Get

To One relationships

The Set and Clear operations are intended to establish and remove a relation between two entities. The name of the operations are composed as `Set<relationship name>` and `Clear<relationship name>`.

See [Entity web service details](#) for more information about representing property values in SOAP requests and responses.

Set operation

SOAP request - Set

Assume that an Order entity has a To One relationship with the BusinessPartner entity, where the relationship is called Customer.

The request to set the customer related to an order takes either the Id or the ItemId of both the Order and the BusinessPartner (if both are provided, the ItemId will be used). An example request that sets the Customer of an Order follows:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <SetCustomer xmlns="http://schemas/OpenTextOrderManagement/Order/operations">
      <ns0:Order-id xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
        <ns0:Id>2</ns0:Id>
      </ns0:Order-id>
      <ns0:Customer xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
        <ns1:BusinessPartner-id
          xmlns:ns1="http://schemas/OpenTextOrderManagement/BusinessPartner">
          <ns1:Id>1</ns1:Id>
        </ns1:BusinessPartner-id>
      </ns0:Customer>
    </SetCustomer>
  </SOAP:Body>
</SOAP:Envelope>
```

This web service is idempotent. If both entities already have a relationship, no errors will be returned.

SOAP response - Set

On successful execution of the operation, an empty SOAP response is returned:

```
<wstxns1:SetCustomerResponse  
xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
```

If an error occurs, a SOAP fault is returned.

Clear operation

SOAP request - Clear

Assume that an Order entity has a To One relationship with the BusinessPartner entity, where the relationship is called Customer.

The request to remove the customer related to an order takes either the Id or the ItemId of the Order entity (if both are provided, the ItemId will be used). An example request that clears the Customer of an Order follows:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">  
  <SOAP:Body>  
    <ClearCustomer xmlns="http://schemas/OpenTextOrderManagement/Order/operations">  
      <ns0:Order-id xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">  
        <ns0:Id>2</ns0:Id>  
      </ns0:Order-id>  
    </ClearCustomer>  
  </SOAP:Body>  
</SOAP:Envelope>
```

This web service is idempotent. If the order does not yet have a relation with a customer, no errors will be returned.

SOAP response - Clear

On successful execution of the operation, an empty SOAP response is returned:

```
<wstxns1:ClearCustomerResponse  
xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
```

If an error occurs, a SOAP fault is returned.

To Many relationships

The AddTo, Get, and RemoveFrom operations are intended to establish, retrieve or remove one or more relationships between two entities. The names of the operations are composed as AddTo<relationship name>, Get<relationship name>, and RemoveFrom<relationship name>.

See [Entity web service details](#) for more information about representing property values in SOAP requests and responses.

AddTo operation

SOAP request - AddTo

Assume that a BusinessPartner entity has a To Many relationship called Orders with the Order entity – where Order has a To One relationship called Customer with Order.

The request to add orders related to a customer takes multiple arguments. It requires the Id or ItemId of the Customer entity and the Ids or ItemIds of all Orders to be related to the Customer (if both Id and ItemId are provided, the ItemId will be used). The following example adds two Orders with ids=1, 2 to the Customer with Id = 1:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <AddToOrders
      xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner/operations">
      <ns0:BusinessPartner-id
        xmlns:ns0="http://schemas/OpenTextOrderManagement/BusinessPartner">
        <ns0:Id>1</ns0:Id>
      </ns0:BusinessPartner-id>
      <ns0:Orders
        xmlns:ns0="http://schemas/OpenTextOrderManagement/BusinessPartner">
        <ns1:Order-id
          xmlns:ns1="http://schemas/OpenTextOrderManagement/Order">
          <ns1:Id>1</ns1:Id>
        </ns1:Order-id>
        <ns1:Order-id
          xmlns:ns1="http://schemas/OpenTextOrderManagement/Order">
          <ns1:Id>2</ns1:Id>
        </ns1:Order-id>
      </ns0:Orders>
    </AddToOrders>
  </SOAP:Body>
</SOAP:Envelope>
```

This web service is idempotent when the To Many relationship is paired with a To One relationship. That is, if a specified Order already has a relationship with the Customer, no errors will be returned.

This web service is not idempotent when the To Many relationship is unpaired, or is paired with another To Many relationship. That is, if a specified Order already has a relationship with the Customer, it will throw an error.

SOAP response - AddTo

On successful execution of the operation, an empty SOAP response is returned:

```
<wstxns1:AddToOrdersResponse
  xmlns:wstxns1="http://schemas/OpenTextOrderManagement/BusinessPartner/operations"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
```

If an error occurs, a SOAP fault is returned.

Get operation

SOAP request - Get

Assume that a Customer entity has a To Many relationship called Orders with the Order entity – where Order has a To One relationship called Customer with Order.

The request to retrieve all orders related to a customer takes one argument, which is the Id or ItemId of the BusinessPartner entity (if both are provided, the ItemId will be used). The following example retrieves the Orders that the BusinessPartner with Id = 1 has created:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <GetOrders
      xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner/operations">
      <ns0:BusinessPartner-id
        xmlns:ns0="http://schemas/OpenTextOrderManagement/BusinessPartner">
          <ns0:Id>1</ns0:Id>
        </ns0:BusinessPartner-id>
      </GetOrders>
    </SOAP:Body>
  </SOAP:Envelope>
```

SOAP response - Get

The response contains the details of each linked order, in the same format as returned by the [Read web service operation](#). An example response follows:

```
<wstxns1:GetOrdersResponse
  xmlns:wstxns1="http://schemas/OpenTextOrderManagement/BusinessPartner/operations"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <wstxns2:Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wstxns1="http://schemas/OpenTextOrderManagement/BusinessPartner/operations"
    xmlns="http://schemas/OpenTextOrderManagement/Order"
    xmlns:wstxns2="http://schemas/OpenTextOrderManagement/Order">
    <Customer xmlns="http://schemas/OpenTextOrderManagement/Order">
      <wstxns3:BusinessPartner-id
        xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner"
        xmlns:wstxns3="http://schemas/OpenTextOrderManagement/BusinessPartner">
        <Id xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">1</Id>
        <ItemId
          xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">00505601024C11E6F0CA5D24D3789644.1</ItemId>
        </wstxns3:BusinessPartner-id>
      </Customer>
      <Order-id xmlns="http://schemas/OpenTextOrderManagement/Order">
        <Id>1</Id>
        <ItemId>00505601024C11E6F0CA5D24D37E3644.1</ItemId>
      </Order-id>
      <ShippingDate xmlns="http://schemas/OpenTextOrderManagement/Order">2015-09-
```

```

14Z</ShippingDate>
    <Amount xmlns="http://schemas/OpenTextOrderManagement/Order">400.0</Amount>
</wstxns2:Order>
<wstxns4:Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wstxns1="http://schemas/OpenTextOrderManagement/BusinessPartner/operations"
xmlns="http://schemas/OpenTextOrderManagement/Order"
xmlns:wstxns4="http://schemas/OpenTextOrderManagement/Order">
    <Customer xmlns="http://schemas/OpenTextOrderManagement/Order">
        <wstxns5:BusinessPartner-id
xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner"
xmlns:wstxns5="http://schemas/OpenTextOrderManagement/BusinessPartner">
            <Id xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">1</Id>
            <ItemId
xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">00505601024C11E6F0CA5D
24D3789644.1</ItemId>
        </wstxns5:BusinessPartner-id>
    </Customer>
    <Order-id xmlns="http://schemas/OpenTextOrderManagement/Order">
        <Id>2</Id>
        <ItemId>00505601024C11E6F0CA5D24D37E3644.2</ItemId>
    </Order-id>
    <ShippingDate xmlns="http://schemas/OpenTextOrderManagement/Order">2015-10-
14Z</ShippingDate>
    <Amount xmlns="http://schemas/OpenTextOrderManagement/Order">500.0</Amount>
</wstxns4:Order>
</wstxns1:GetOrdersResponse>

```

RemoveFrom operation

SOAP request - RemoveFrom

Assume that a Customer entity has a To Many relationship with the Order entity – where Order has a To One relationship called Customer with Order.

The request to remove orders related to a customer takes multiple arguments. It requires the Id or ItemId of the Customer entity and the Ids of all Orders that have a relation to the Customer, for which the relation has to be removed (if both Id and ItemId are provided, the ItemId will be used). The following example removes the relations of two Orders with the Customer with Id = 1:

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP:Body>
        <RemoveFromOrders
xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner/operations">
            <BusinessPartner-id
xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">
                <Id>1</Id>
            </BusinessPartner-id>
            <Orders xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">
                <Order-id xmlns="http://schemas/OpenTextOrderManagement/Order">
                    <Id>1</Id>

```

```
</Order-id>
</Orders>
<Orders xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">
    <Order-id xmlns="http://schemas/OpenTextOrderManagement/Order">
        <Id>3</Id>
    </Order-id>
</Orders>
</RemoveFromOrders>
</SOAP:Body>
</SOAP:Envelope>
```

This web service is idempotent when the To Many relationship is paired with a To One relationship. That is, if any specified Order does not have a relationship with the Customer, no errors will be returned.

This web service is not idempotent when the To Many relationship is unpaired, or is paired with another To Many relationship. That is, if any Order does not have a relation with the Customer, no errors will be returned.

SOAP response - RemoveFrom

On successful execution of the operation, an empty SOAP response is returned.

```
<wstxns1:RemoveFromOrdersResponse
xmlns:wstxns1="http://schemas/OpenTextOrderManagement/BusinessPartner/operations"
mlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
```

If an error occurs, a SOAP fault is returned.

To Child relationships

The create, get, and delete child operations are intended to create, retrieve, and delete one or more children of a parent entity. The name of the operation is composed as Create<relationship_name>, Get<relationship_name>, and Delete<relationship_name>.

In the following example SOAP messages, the parent entity is an Order having a parent child relationship called OrderLine to child entity OrderLine.

Note: For parent child relationships, it is required to make the name of the relationship the same as the name of the child entity, in this example it is OrderLine.

See [Entity web service details](#) for more information about representing property values in SOAP requests and responses.

Create operation

SOAP request - Create

Assume that Order with Id=16385 has been created with CreateOrder.

The request takes either the Id or the ItemId of an entity (if both are provided, the ItemId will be used).

The following example creates an OrderLine in Order Id=16385, it also assigns OrderLine with properties Product and Color. It is possible to create more than one OrderLine in the same request. Properties in the request can be set only when **Allow participants to change the value** is set to **Anytime** or **Only when first created**. See [Adding properties](#).

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <CreateOrderLine xmlns="http://schemas/OpenTextOrderManagement/Order/operations">
      <ns0:Order-id xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
        <ns0:Id>16385</ns0:Id>
      </ns0:Order-id>
      <ns1:OrderLine-create
        xmlns:ns1="http://schemas/OpenTextOrderManagement/Order.OrderLine">
        <ns1:Product>Ball</ns1:Product>
        <ns1:Color>Green</ns1:Color>
      </ns1:OrderLine-create>
    </CreateOrderLine>
  </SOAP:Body>
</SOAP:Envelope>
```

SOAP response - Create

The response contains the details of the created OrderLine. In response it shows that OrderLine is created with Id1=1, where Id1 is the OrderLine part. The response contains all the properties of the updated entity, independent of the **Allow participants to change the value** setting.

```
<wstxns1:CreateOrderLineResponse
  xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <wstxns2:OrderLine
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
    xmlns="http://schemas/OpenTextOrderManagement/Order.OrderLine"
    xmlns:wstxns2="http://schemas/OpenTextOrderManagement/Order.OrderLine">
    <OrderLine-id
      xmlns="http://schemas/OpenTextOrderManagement/Order.OrderLine">
      <Id>16385</Id>
      <Id1>1</Id1>
      <ItemId>00505601024C11E6F0CA5D24D37E3644.16385</ItemId>
      <ItemId1>00505601024C11E6F0CBA6E20481B644.16385.1</ItemId1>
    </OrderLine-id>
    <Product
      xmlns="http://schemas/OpenTextOrderManagement/Order.OrderLine">Ball</Product>
    <Color
      xmlns="http://schemas/OpenTextOrderManagement/Order.OrderLine">Green</Color>
  </wstxns2:OrderLine>
</wstxns1:CreateOrderLineResponse>
```

Get operation

SOAP request - Get

Assume that an Order with Id=16385 with OrderLine of Id1=1 are created.

To get the OrderLines from Order it needs the Order Id or ItemId (if both are provided, the ItemId will be used). The following example retrieves all OrderLines of Order with Id=16385.

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <GetOrderLine xmlns="http://schemas/OpenTextOrderManagement/Order/operations">
      <ns0:Order-id xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
        <ns0:Id>16385</ns0:Id>
      </ns0:Order-id>
    </GetOrderLine>
  </SOAP:Body>
</SOAP:Envelope>
```

SOAP response - Get

The response contains the details of all OrderLines of Order. In the following response OrderLine Id1=1 of Order Id=16385 is returned, including the Orderline properties:

```
<wstxns1:GetOrderLineResponse
  xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <wstxns2:OrderLine xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
    xmlns="http://schemas/OpenTextOrderManagement/Order.OrderLine"
    xmlns:wstxns2="http://schemas/OpenTextOrderManagement/Order.OrderLine">
    <OrderLine-id xmlns="http://schemas/OpenTextOrderManagement/Order.OrderLine">
      <Id>16385</Id>
      <Id1>1</Id1>
      <ItemId>00505601024C11E6F0CA5D24D37E3644.16385</ItemId>
      <ItemId1>00505601024C11E6F0CBA6E20481B644.16385.1</ItemId1>
    </OrderLine-id>
    <Product
      xmlns="http://schemas/OpenTextOrderManagement/Order.OrderLine">Ball</Product>
    <Color
      xmlns="http://schemas/OpenTextOrderManagement/Order.OrderLine">Green</Color>
  </wstxns2:OrderLine>
</wstxns1:GetOrderLineResponse>
```

Delete operation

The delete operation provides the capability to delete children or compositely owned entities. One entity can be deleted per request.

SOAP request - Delete

To delete one or more OrderLines from Order, the Order needs to be identified, which is the first parameter (Id=16385). And then to be deleted OrderLines are identified with of Order Id and OrderLine Id1. It is possible to provide the properties of the OrderLine too but this is optional.

The request takes either the Id or the ItemId of the entities (if both are provided, the ItemId will be used).

Assume that an Order with Id=16385 has three OrderLines (with Id1's 1, 2 and 3). The following example deletes OrderLine with Id1=1 from the Order with Id=16385.

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <DeleteOrderLine xmlns="http://schemas/OpenTextOrderManagement/Order/operations">
      <ns0:Order-id xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
        <ns0:Id>16385</ns0:Id>
      </ns0:Order-id>
      <ns1:OrderLine
        xmlns:ns1="http://schemas/OpenTextOrderManagement/Order.OrderLine">
        <ns1:OrderLine-id>
          <ns1:Id>16385</ns1:Id>
          <ns1:Id1>1</ns1:Id1>
        </ns1:OrderLine-id>
        <ns1:Product>Ball</ns1:Product>
        <ns1:Color>Green</ns1:Color>
      </ns1:OrderLine>
    </DeleteOrderLine>
  </SOAP:Body>
</SOAP:Envelope>
```

Note: The Product and Color properties are passed for feeding the conflict detection. See [When users work on this property at the same time, report conflicts](#) for more information.

SOAP response - Delete

The response of DeleteOrderLine is empty.

```
<wstxns1:DeleteOrderLineResponse
  xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
```

To Parent relationship

The get operation retrieves the parent entity from a child entity. The name of the operation is composed as Get<parent entity name>.

In the following SOAP messages, the parent entity is an Order having a parent child relationship called Orderline to child entity OrderLine, and the Orderline has To Parent relationship Order back to the Order entity.

See [Entity web service details](#) for more information about representing property values in SOAP requests and responses.

Get operation

SOAP request - Get

Assume that an Order with Id=16385 with OrderLines of Id1=1,2,3 is created. Either the Id or the ItemId of the entities can be used (if both are provided, the ItemId will be used)

To get the parent entity Order from a child entity OrderLine, it needs the Order Id, and the OrderLine Id1. The following example retrieves Order (Id=16385) from OrderLine (Id1=2):

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <GetOrder
      xmlns="http://schemas/OpenTextOrderManagement/Order.OrderLine/operations">
      <ns0:OrderLine-id
        xmlns:ns0="http://schemas/OpenTextOrderManagement/Order.OrderLine">
          <ns0:Id>16385</ns0:Id>
          <ns0:Id1>2</ns0:Id1>
        </ns0:OrderLine-id>
      </GetOrder>
    </SOAP:Body>
  </SOAP:Envelope>
```

SOAP response - Get

The response contains the details of the parent Order entity. In the following response, Order (Id=16385), including its properties, is returned:

```
<wstxns1:GetOrderResponse
  xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order.OrderLine/operations"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <wstxns2:Order
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order.OrderLine/operations"
    xmlns="http://schemas/OpenTextOrderManagement/Order"
    xmlns:wstxns2="http://schemas/OpenTextOrderManagement/Order">
    <Order-id
      xmlns="http://schemas/OpenTextOrderManagement/Order">
      <Id>16385</Id>
      <ItemId>00505601024C11E6F0CA5D24D37E3644.16385</ItemId>
    </Order-id>
    <ShippingDate
      xmlns="http://schemas/OpenTextOrderManagement/Order">2015-09-14Z</ShippingDate>
    <Amount
      xmlns="http://schemas/OpenTextOrderManagement/Order">300.0</Amount>
  </wstxns2:Order>
</wstxns1:GetOrderResponse>
```

Configuring a Find web service operation

A Find web service operation is configured with the expression editor. The expression editor is shown in the Filter tab. See [Using the expression editor](#)

For the Find web service, the following modifications have been made to the expression editor:

- The Advanced tab is hidden to safeguard that the entity modeler creates expressions that can be executed on a database.
- Because the Advanced tab is hidden, the Basic tab is not shown.
- The browse list for properties on the Basic tab is extended with a **Browse** option to navigate to properties of related entities.
- A Parameter and a Results tab were added next to the Filter tab to enable the definition of parameters and sorting for the web service operation.
- In the Basic tab, you can specify for most operations whether a static value or a parameter (defined in the Parameters tab) must be used.

Note: If an invalid expression is created (for example, when the data types of a property and a parameter do not match), the expression is shown in the (normally hidden) Advanced tab of the expression editor so that it can be corrected there. If it is valid, switching to the Parameter tab and back to the Filter tab shows the expression in the Basic tab again.

All parameters in the Parameter tab must be used in the Filter tab or an error is generated at validation or publication of the entity.

SOAP request – Find

Assume that a BusinessPartner entity, having a property called Name, has a To Many relationship called Orders with the Order entity, having a property called OrderDate – where Order has a To One relationship back with BusinessPartner called Customer.

You can configure a Find web service operation called FindOrdersByCustomerName on BusinessPartner as follows:

- In the Parameter tab add a parameter called BusinessPartnerName of data type Text
- In the Filter tab add the following filter:
item.Customer.Properties.Name - equal to - Parameter – BusinessPartnerName
- In the Results tab sort the results in Descending order by item.Properties.OrderDate

The generated Find request will look as follows:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <FindOrdersByCustomerName xmlns="http://schemas/CP/Order/operations">
      <BusinessPartnerName>PARAMETER</BusinessPartnerName>
      <ns0:Cursor xmlns:ns0="http://schemas.opentext.com/bps/entity/core" offset="0"
limit="100" />
    </FindOrdersByCustomerName>
  </SOAP:Body>
</SOAP:Envelope>
```

The specified parameter BusinessPartnerName is shown as a tag in the request.

The Cursor tag specifies how many orders have to be returned at max (in the attribute called limit) and with which order to start (in the attribute called offset).

SOAP response – Find

If the previous request is fired for a business partner, with name 'ACME', having 2 orders, the response will look as follows:

```
<wstxns1:FindOrdersByCustomerNameResponse
    xmlns:wstxns1="http://schemas/CP/Order/operations"
    xmlns:wstxns2="http://schemas/CP/Order"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<wstxns2:Order>
    <Order-id xmlns="http://schemas/CP/Order">
        <Id>1</Id>
        <ItemId>00505601024CA1E7A6195BE99E365458.1</ItemId>
    </Order-id>
    <Customer xmlns="http://schemas/CP/Order">
        <BusinessPartner-id xmlns="http://schemas/CP/BusinessPartner">
            <Id>1</Id>
            <ItemId>00505601024CA1E7A6195BE99E2C5458.1</ItemId>
        </BusinessPartner-id>
    </Customer>
    </wstxns2:Order>
    <wstxns2:Order>
        <Order-id xmlns="http://schemas/CP/Order">
            <Id>2</Id>
            <ItemId>00505601024CA1E7A6195BE99E365458.2</ItemId>
        </Order-id>
        <Customer xmlns="http://schemas/CP/Order">
            <BusinessPartner-id xmlns="http://schemas/CP/BusinessPartner">
                <Id>1</Id>
                <ItemId>00505601024CA1E7A6195BE99E2C5458.1</ItemId>
            </BusinessPartner-id>
        </Customer>
    </wstxns2:Order>
</wstxns1:FindOrdersByCustomerNameResponse>
```

If the business partner has more than the number of orders specified in `limit` the response would also contain a cursor tag such as the following:

```
<Cursor xmlns ="http://schemas.opentext.com/bps/entity/core" limit="100"
offset="100" />
```

You can paste this tag into the request to get the next chunk of 100 orders for this customer.

Note: You can sort the results in the web service response either in ascending or descending order by specifying the sort order for one of the properties of the entity.

Web services on other building blocks

The previous examples show that the entity web service contains the entity's identity, its properties, and its relationships. However, the web service can also contain the properties of other building blocks. The properties of these building blocks are included in the entity requests and responses, depending on whether the property is read-only.

For example, if an entity has a Title building block, this is added to the entity create request.

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <CreateOrder xmlns="http://schemas/OpenTextOrderManagement/Order/operations">
      <ns0:Order-create xmlns:ns0="http://schemas/OpenTextOrderManagement/Order">
        <ns0:Amount>400.0</ns0:Amount>
        <ns0:ShippingDate>2015-09-14Z</ns0:ShippingDate>
        <ns2:Title-create
          xmlns:ns2="http://schemas.opentext.com/entitymodeling/buildingblocks/title/1.0">
          <ns2:Title>Order for Johnson</ns2:Title>
        </ns2:Title-create>
      </ns0:Order-create>
    </CreateOrder>
  </SOAP:Body>
</SOAP:Envelope>
```

The response will appear as follows:

```
<wstxns1:CreateOrderResponse
  xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <wstxns2:Order
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wstxns1="http://schemas/OpenTextOrderManagement/Order/operations"
    xmlns="http://schemas/OpenTextOrderManagement/Order"
    xmlns:wstxns2="http://schemas/OpenTextOrderManagement/Order">
    <wstxns3:Title
      xmlns="http://schemas.opentext.com/entitymodeling/buildingblocks/title/1.0"
      xmlns:wstxns3="http://schemas.opentext.com/entitymodeling/buildingblocks/title/1.0">
      <Title
        xmlns="http://schemas.opentext.com/entitymodeling/buildingblocks/title/1.0">Order for
      Johnson</Title>
    </wstxns3:Title>
    <Order-id
      xmlns="http://schemas/OpenTextOrderManagement/Order">
      <Id>32769</Id>
      <ItemId>00505601024C11E6F0CA5D24D37E3644.32769</ItemId>
    </Order-id>
    <ShippingDate
      xmlns="http://schemas/OpenTextOrderManagement/Order">2015-09-
    14Z</ShippingDate>
    <Amount
      xmlns="http://schemas/OpenTextOrderManagement/Order">400.0</Amount>
  </wstxns2:Order>
</wstxns1:CreateOrderResponse>
```

Similarly, for the UpdateOrder operation a <Title-update> element is added, containing the properties that can be updated.

Other building blocks expose their properties in a way similar to the Title building block.

Entity web service details

In the request and response of a web service operation, the entity is represented in XML. This topic describes various details with respect to the XML.

Entity representation

The following XML fragment shows an example of how an entity is represented in XML. The entity is an Order that has 3 user defined properties: Customer, Amount, and ShippingDate.

```
<Order xmlns="http://schemas/packagename/Order">
  <Order-id>
    <Id>1</Id>
  </Order-id>
  <Customer>Johnson</Customer>
  <Amount>500.0</Amount>
  <ShippingDate>2015-04-21</ShippingDate>
</Order>
```

The following sections describe each part of the XML.

Root node

The root node of the XML is as follows.

```
<Order xmlns="http://schemas/packagename/Order">
```

The tag name is equal to the Entity name. The value of the namespace is described [here](#).

Identity

The identity is a piece of XML, with tag name: <name of the entity>-id. For entity Order it will look like:

```
<Order-id>
  <Id>1</Id>
</Order-id>
```

For child entities, the parent Id is added to the Id of the entity. For child entity OrderLine, having a parent Order with Id=1 it will look as follows.

```
<OrderLine-id>
```

```
<Id>1</Id>    <---- parent Id
<Id1>1</Id1>
</OrderLine-id>
```

Responses also expose internal information in the id: ItemId. For child entity OrderLine, having a parent Order with Id=1 it will appear as follows.

```
<OrderLine-id>
  <Id>1</Id>    <---- parent Id
  <Id1>1</Id1>
  <ItemId>005056871e6211e5eac00020682b5ba9.1</ItemId> <-- parent ItemId
  <ItemId1>005056871e6211e5eac023adb9ec5ba2.1.1</ItemId1>
</OrderLine-id>
```

Property value

An example of a user defined property Name follows.

```
<Name>John</Name>
```

The tag name is the name of the property. The text content is its value.

Empty value

An empty value is an XML node with no text content. For example:

```
<Name></Name>
```

If the property is of type String, it is an empty string (""). Empty text content is not allowed for other types, such as Boolean and Decimal. If you want to express that the value is undefined, give it the null value by using `xsi:nil`.

Null value

A null value is represented using the `xsi:nil` attribute, as follows.

```
<Name xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
```

If `xsi:nil` has the value "true", any text content of the node is ignored.

Date value

A Date value is represented with the format `yyyy-mm-ddZ`. The trailing Z is the zone designator for the zero UTC offset time zone. For example:

```
<OrderDate>2015-04-22Z</OrderDate>
```

Duration value

A Duration value is represented with the format PxxYxxMxxWxxDTxxHxxMxxS (in accordance with [ISO-8601](#)).

- P is the duration designator (historically called "period") placed at the start of the duration representation.
- Y is the year designator that follows the value for the number of years.
- M is the month designator that follows the value for the number of months.
- W is the week designator that follows the value for the number of weeks.
- D is the day designator that follows the value for the number of days.
- T is the time designator that precedes the time components of the representation.
- H is the hour designator that follows the value for the number of hours.
- M is the minute designator that follows the value for the number of minutes.
- S is the second designator that follows the value for the number of seconds.

For example a duration of 10 days, 2 hours, and 3 minutes is specified as follows:

```
<Duration>P10DT2H3M</Duration>
```

Date and Time value

A Date and Time value is represented with the format yyyy-mm-ddThh:mm:ssZ. The trailing Z is the zone designator for the zero UTC offset time zone. For example:

```
<DeliveryTime>2015-04-22T10:28:12Z</DeliveryTime>
```

XML Namespace

The following sections describe the value of the namespace.

Entity namespace

The namespace of an entity is composed of different parts, separated by a slash (/).

```
http://schemas/<packagename>/<entityname>
```

- `http://schemas` is a fixed namespace prefix.
- `<packagename>` is the Package Name as specified in the package properties.
- `<entityname>` is the Name of the entity.

Note: The Package Name is defined in CWS in the package properties, and can be viewed or edited by opening the project in **Workspace Documents** and then choosing Packaging > Package Properties in the shortcut menu. The Package Owner and Product Name are shown as fields in the dialog box.

The name of a child entity is composed by the name(s) of its parent(s) and the name of the child entity itself, separated by a dot (.). That is, when having an Order entity with OrderLine as a child, the namespace will be `http://schemas/<packagename>/Order.OrderLine`.

Operation namespace

The namespace from a web service operation is equal to the namespace of the entity, extended with `/operations`. For example:

```
http://schemas/<packagename>/<entityname>/operations
```

The namespace is generated from CWS project properties. If these properties change, the namespace changes accordingly.

Note: As soon as a web service operation is used in a model (such as the User Interface or BPM), the namespace is copied into that model. Changing the Package Name changes the namespace. You must then re-bind the web service operation to the model.

Reserved characters

The XML namespace is a URI that has certain restrictions with respect to which characters are allowed.

Unreserved characters: Alphabetic, decimal digits, - _ . ! ~ * ' ()

Reserved characters: All other characters. They are encoded using '%'. e.g. "\$" becomes %24.

If the solution name contains reserved characters, they are encoded.

Examples

Package owner	Project name	Solution name
MyCompany	MyProject	MyCompanyMyProject
My Company	ProjectA	My%20CompanyProjectA
The \$\$ Bank	Insurance	The%20%24%24%20BankInsurance

Subtype entity namespace

An entity can be subtyped. A subtype has its own namespace to distinguish it from its supertype. In the XML representation:

- The subtype node has the namespace of the subtype
- The properties and relationships of the subtype have the namespace of the subtype
- The properties and relationships of the supertype have the namespace of the supertype

Assume that an entity called Order includes properties called Customer and Amount. The entity SalesOrder is created as a subtype of Order, adding the property Discount and the relationship SalesRep. The XML representation of a SalesOrder appears as follows:

```
<SalesOrder xmlns="http://schemas/packagename/SalesOrder">
  <Order-id xmlns="http://schemas/packagename/Order">
    <Id>1</Id>
  </Order-id>
  <Customer xmlns="http://schemas/packagename/Order">Johnson</Customer>
  <Amount xmlns="http://schemas/packagename/Order">500.0</Amount>
  <Discount>10</Discount>
  <SalesRep>
    <Employee-id xmlns="http://schemas/packagename/Employee">
      <Id>1</Id>
    </Employee-id>
  </SalesRep>
</SalesOrder>
```

The subtype entity has the namespace of SalesOrder. The Order-id is inherited from the supertype, so it has the namespace of Order, just like the properties Customer and Amount. The property Discount and the relationship SalesRep (defined on SalesOrder) have the namespace of SalesOrder.

In the SOAP requests for SalesOrder, the different namespaces are also used. The read operation for example appears as follows:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <ReadSalesOrder xmlns="http://schemas/packagename/SalesOrder/operations">
      <Order-id xmlns="http://schemas/packagename/Order">
        <Id>1</Id>
      </Order-id>
    </ReadOrder>
  </SOAP:Body>
</SOAP:Envelope>
```

The namespace of the read operation is based on SalesOrder. The Order-id is inherited from the supertype, so it is in the namespace of Order.

Customization namespace

An entity can be customized. The impact on the XML representation is similar to the impact of subtyping:

- The custom entity node has the namespace of the customization.
- The properties and relationships of the customized entity have the namespace of the customization.
- The properties and relationships of the original entity have the namespace of the original entity.

SOAP web services on Process Experience

This section provides information about implemented SOAP web services. The functionality of the SOAP web services is based on Process Experience.

Email web services

This section provides information about implemented SOAP web services for Email and Email Template capabilities.

sendEmail

The sendEmail web service is used to send an Email through to Process Experience and attach it to a particular item.

SOAP request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP: Body>
    <sendEmail xmlns="http://schemas.cordys.com/entity/email/1.0">
      <ItemId>PARAMETER</ItemId>
      <Email>
        <to>
          <address>PARAMETER</address>
        </to>
        <cc>
          <address>PARAMETER</address>
        </cc>
        <subject>PARAMETER</subject>
        <body>PARAMETER</body>
        <attachments>
          <attachment>PARAMETER</attachment>
        </attachments>
      </Email>
    </sendEmail>
  </SOAP: Body>
</SOAP:Envelope>
```

Request parameters

Parameter	Description	Data Type	Required
ItemId	ItemId of the item	String	Yes
address	Any email address	String	To is required Cc is Optional
subject	Subject of the email	String	Yes
body	Body of the email	String	Optional

Parameter	Description	Data Type	Required
attachment	Document Id that exists in the Content panel	String	Optional

SOAP response

```
<sendEmailResponse xmlns="http://schemas.cordys.com/entity/email/1.0"/>
```

Note: The message mapping request should be in the same order as defined above.

sendEmailFromTemplate

The sendEmailFromTemplate web service is used to send an Email using an existing Email Template through to Process Experience and attach it to a particular item.

SOAP request

```
<SOAP: Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP: Body>
        <sendEmailFromTemplate xmlns="http://schemas.cordys.com/entity/email/1.>
            <ItemId>PARAMETER</ItemId>
            <Email>
                <templateId>PARAMETER</templateId>
                <to>
                    <address>PARAMETER</address>
                </to>
                <cc>
                    <address>PARAMETER</address>
                </cc>
                <subject>PARAMETER</subject>
                <body>PARAMETER</body>
                <attachments>
                    <attachment>PARAMETER</attachment>
                </attachments>
            </Email>
        </sendEmailFromTemplate>
    </SOAP: Body>
</SOAP: Envelope>
```

SOAP response

```
<sendEmailFromTemplateResponse xmlns="http://schemas.cordys.com/entity/email/1.0"/>
```

Request parameters

Parameter	Description	Data Type	Required
ItemId	ItemId of the item	String	Yes
templateId	Template Id that exists in the entity		Yes
address	Any email address	String	To is required (if not provided in the template) Cc is Optional
subject	Subject of the email	String	Yes (if not provided in the template)
body	Body of the email	String	Optional
attachment	Document Id that exists in the Content panel	String	Optional

Note:

- Message mapping Request should be in the same order as defined above.
- If you have not provided any values in the above request it will take template values.

getEmailTemplate

The getEmailTemplate web service is used to get a particular email template that is available in an entity.

SOAP request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP: Body>
    <getEmailTemplate xmlns="http://schemas.cordys.com/entity/email/1.0">
      <ItemId>PARAMETER</ItemId>
      <TemplateID>Parameter</TemplateID>
    </getEmailTemplate>
  </SOAP: Body>
</SOAP:Envelope>
```

SOAP response

```
<getEmailTemplateResponse xmlns="http://schemas.cordys.com/entity/email/1.0">
  <EmailTemplate>
    <TemplateName>PARAMETER</TemplateName>
    <To>PARAMETER</To>
```

```
<Cc>PARAMETER</Cc>
<Subject>PARAMETER</Subject>
<Body>PARAMETER</Body>
</EmailTemplate>
</getEmailTemplateResponse>
```

Request parameters

Parameter	Description	Data Type	Required
ItemId	ItemId of the item	String	Yes
templateId	Template Id that exists in the entity	String	Yes

getAllEmailTemplates

The getAllEmailTemplates web service is used to get all the email templates that are available in a particular entity.

SOAP request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP: Body>
    <getAllEmailTemplates xmlns="http://schemas.cordys.com/entity/email/1.0">
      <ItemId>PARAMETER</ItemId>
    </getAllEmailTemplates>
  </SOAP: Body>
</SOAP:Envelope>
```

SOAP response

```
<getAllEmailTemplatesResponse xmlns="http://schemas.cordys.com/entity/email/1.0">
  <EmailTemplates>
    <EmailTemplate>
      <TemplateID>PARAMETER</TemplateID>
      <TemplateName>PARAMETER</TemplateName>
      <To>PARAMETER</To>
      <Cc>PARAMETER</Cc>
      <Subject>PARAMETER</Subject>
      <Body>PARAMETER</Body>
    </EmailTemplate>
  </EmailTemplates>
</getAllEmailTemplatesResponse>
```

Request parameters

Parameter	Description	Data Type	Required
ItemId	ItemId of the item	String	Yes

File and Content web services

This section provides information about implemented SOAP web services for File and Content capabilities.

UploadDocument

The UploadDocument web service is used to perform the following tasks in Process Experience:

- Upload an empty document record.
- Upload a document.
- Upload an updated document on top of an existing document.

SOAP request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <UploadDocument
      xmlns="http://schemas.opentext.com/bps/entity/buildingblock/file">
      <Create>
        <ns0:ItemId
          xmlns:ns0="http://schemas.opentext.com/bps/entity/core">PARAMETER</ns0:ItemId>
          <Title>PARAMETER</Title>
          <FileName>PARAMETER</FileName>
          <FileURL>PARAMETER</FileURL>
          <FolderPath>PARAMETER</FolderPath>
        </Create>
        <Update>
          <ns0:ItemId
            xmlns:ns0="http://schemas.opentext.com/bps/entity/core">PARAMETER</ns0:ItemId>
            <Title>PARAMETER</Title>
            <FileName>PARAMETER</FileName>
            <FileURL>PARAMETER</FileURL>
          </Update>
        </UploadDocument>
      </SOAP:Body>
    </SOAP:Envelope>
```

Request parameters

Parameter	Description	Data Type
ItemId	ItemId of the item	String

Parameter	Description	Data Type
Title	Title to be shown in the Content panel	String
FileName	Name of the file to selected for upload	String
FileURL	URL of the file location	String
FolderPath	Folder to which the file will be uploaded on the Content panel	String

SOAP response

```
<data>
  <ns2:UploadDocumentResponse
  xmlns:ns2="http://schemas.opentext.com/bps/entity/buildingblock/file"
  xmlns="http://schemas.opentext.com/bps/entity/core">
    <ItemId xmlns:ns2="http://schemas.opentext.com/bps/entity/buildingblock/file"
    xmlns="http://schemas.opentext.com/bps/entity/core">parameter</ItemId>
  </ns2:UploadDocumentResponse>
</data>
```

Examples

Following is an example payload for creating a placeholder record in a folder.

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <UploadDocument
    xmlns="http://schemas.opentext.com/bps/entity/buildingblock/file">
      <Create>
        <ns0:ItemId
        xmlns:ns0="http://schemas.opentext.com/bps/entity/core">00505681086711E7EADFA772E20ED
        0D2.32769</ns0:ItemId>
        <Title>WorkExperience</Title>
        <FolderPath>My Documents/Employment History</FolderPath>
      </Create>
    </UploadDocument>
  </SOAP:Body>
</SOAP:Envelope>
```

Important: Do not pass `FileName` when you want to create a placeholder record.

Following is an example payload for uploading a document in a folder.

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <UploadDocument
    xmlns="http://schemas.opentext.com/bps/entity/buildingblock/file">
      <Create>
```

```

<ns0:ItemId
xmlns:ns0="http://schemas.opentext.com/bps/entity/core">00505681086711E7EADFA772E20ED
0D2.32769</ns0:ItemId>
<Title>MyTitle</Title>
<FileName>workspace.png</FileName>

<FileURL>http://uk07484.opentext.net:8080/home/system/wcp/theme/default/image/wizard/
workspace.png</FileURL>
<FolderPath>My Documents/Employment History</FolderPath>
</Create>
</UploadDocument>
</SOAP:Body>
</SOAP:Envelope>

```

Following is an example payload for updating a document in a folder.

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <UploadDocument
      xmlns="http://schemas.opentext.com/bps/entity/buildingblock/file">
      <Update>
        <ns0:ItemId
          xmlns:ns0="http://schemas.opentext.com/bps/entity/core">00505681086711E7EAE01170AAEB3
          0D2.32769.32774</ns0:ItemId>
        <Title>DocumentTitle</Title>
        <FileName>finish.png</FileName>

        <FileURL>http://uk07484.opentext.net:8080/home/system/wcp/theme/default/image/wizard/
        finish.png</FileURL>
      </Update>
      </UploadDocument>
    </SOAP:Body>
  </SOAP:Envelope>

```

Setting deadlines

The Deadline building block specifies the schedule for completing an item. A deadline runs based on attributes that you specify. For example, you specify when to start and stop the policy and define the schedule as a specified number of days, hours, and minutes or as a value that is read from a property. You can also define actions to trigger when the deadline is approaching or missed. This enables you to automate reminders, warnings, and actions that are aimed at keeping processes on track to meet deadlines.

Upon adding a deadline, a property called `<DeadlineName>_DueDate` with a type of `datetime` is created. This property is shown in the Security Editor with `Update` permission. This permission is selected or cleared to enable or prevent a Process Experience user from setting or updating the property value. The number of properties shown under this section are based on the number of deadlines added to the entity. See [Configuring security](#).

The deadline policy works as follows:

- When you create a deadline, a property called <**DeadlineName**>_**DueDate** with a type of **datetime** is created. The value of this property can be set using rules, web services, or forms. Process Experience users can edit the value of the due date. For example, if a claims manager changes the due date, changes to the deadline are automatically calculated.
- Deadline execution starts when the start policy condition is satisfied.
- When the deadline execution starts, the due date is calculated based on the configured default duration (either static duration or a duration property). If a duration is not specified, a due date is not calculated. The due date can also be specified for a particular item in Process Experience.
- Based on the due date, all deadline actions are scheduled based on the configured action duration and conditions. For each deadline action, when the scheduled time is reached, the configured action (such as sending an e-mail or triggering a process) is executed. If the due date changes, all the configured deadline actions (escalations) are rescheduled based on the modified due date.
- Based on the due date, the scheduled time for all configured deadline actions (escalations) is calculated, and escalations are scheduled if the scheduled time is greater than the time when the due date was created.
- If the due date is modified, all existing scheduled pending escalations are made obsolete and the scheduled time for all configured deadline actions is calculated based on the modified due date. Escalations are scheduled if the scheduled time is greater than the time when the due date was modified.
- Deadline execution terminates when the stop policy condition is satisfied. When deadline execution stops, all pending deadline actions (which are yet to be done) become obsolete and the deadline cycle is complete. A new deadline execution cycle is initiated if the start policy condition is satisfied after the completion of the deadline cycle. Deadline actions and stop policy conditions are also considered for this new deadline execution cycle.
- It is possible to change properties (such as start policy, stop policy, duration, and configured actions) of a deadline whose instances are already running. Changing the properties of a deadline affects runtime instances as follows:
 - Changes to start policy, duration, and deadline actions have no effect on running instances but they are considered for subsequent instances.
 - Changes to the stop policy are considered for both running instances and subsequent instance.

Note: For a deadline Send Email action, you can select an email template that was defined on the entity. In Process Experience email is sent to the To and CC recipients specified in the template. See [Creating an email template](#).

You can configure multiple schedules for an entity that contains the Deadline building block.

To add a Deadline:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Deadline**.

3. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

4. Click **Configure**.

5. On the **Deadline definition** tab:

- a. In **Start policy**, select one of the following options for starting the schedule:
 - **when an instance is created**. Select this option if you want to start the deadline monitoring when an item is created.
 - **when an instance enters a certain state**. This option is available only if the Lifecycle building block was added to the entity. Select this option if you want to start deadline monitoring when a specific lifecycle state starts. When you select this option, all the states in the lifecycle are listed so that you can select the state.
 - **when a certain condition is met**. Select this option if you want to define a custom condition to start deadline monitoring. Deadline execution starts when the start policy condition is satisfied or any time a property that is referenced in the condition is changed. When you select this option, a zoom icon appears. Click the zoom icon to open the expression editor and define the condition. See [Using the expression editor](#).
- b. In **Stop policy**, select one of the following options for stopping the schedule:
 - **when an instance is completed**. This option is available only if the Lifecycle building block was added to the entity. Select this option if you want the deadline monitoring to stop on completion of the lifecycle execution.
 - **when an instance exits a certain state**. This option is available only if **when an instance enters a certain state** is selected in the start policy and cannot be changed (disabled). In this case, the stop condition is defined based on the selected state exit. That is, on the selected state exit, the deadline monitoring will be stopped.
 - **when a certain condition is met**. Select this option if you want to define a custom condition to stop deadline monitoring. Deadline execution stops when the stop policy condition is satisfied or any time a property that is referenced in the condition is changed. When you select this option, a zoom icon appears. Click the zoom icon to open the expression editor and define the condition. See [Using the expression editor](#).
- c. In **Default deadline**, select **Static** or **Read from property**.
 - If you select **Static**, specify the number of days, hours, and minutes.
 - If you select **Read from property**, select the property.

6. Select the **Alerts and escalations** tab and click **Add action**.

The Add Action dialog box opens.

7. In **Add action**, select **Send email**.

8. In **Email template**, click  (Browse) and select the email template.
The To and CC details are filled in as follows:
 - If **From template** is selected (this is the default), email addresses from the selected email template are shown in the To and Cc fields and cannot be edited.
 - If **Custom** is selected, the email addresses in the To and CC fields can be changed.
9. In **Schedule**, type the number of Days, Hours, or Minutes and select **Before deadline** or **After deadline**.
For example, you might want an email to be sent two days before the deadline.
10. Click **Save**.
11. Close the window.

Important: For each deadline cycle, a process (BPM) called **Entity Deadline Monitoring Process** runs in the background in Process Platform. Any actions such as pause, terminate, and restart to the instances of this process will lead to unexpected errors in deadline execution. This process is not visible to Process Experience users but a user with the Administrator role can view it from the Process Instance Manager. See the Process Platform Product Documentation for details.

Caution: In forms, an entity with a Deadline building block displays a relationship called DeadlineInstance in the Components panel. This is an internal relationship that should not be used when designing a form.

Chapter 6

Designing the presentation

The presentation defines the user interface for the application.

Adding layouts

Use the Layout building block to create customized user interfaces for your solutions.

Authorized users can also create their own customized personal user interfaces.

Layouts organize a set of panels into a page to be presented to a Process Experience user.

There are two types of layouts:

- Item layouts
- Home page layouts

A layout can include any number of predefined panels that you arrange into zones. If you combine multiple panels in a single zone, the user interface presents tabs to select a panel. Layouts are designed to present users with the information they need in a way that is optimized for the interactions they are expected to perform. Many applications do not include a home page layout, relying on the system provided default layouts to serve their needs. Similarly, many entities rely on the default item layout, or provide a single layout that works well for all users. Typically, however, you will create multiple application and item layouts, each optimized for a different task or type of user.

For example, an Intake Processor's interactions with a case are quite different from an Underwriter's interactions. The Intake Processor needs to review the information submitted by a customer while the Underwriter needs to be able see the entire case status, access other systems, and interact with support personnel while making a decision about the case. There are several ways to optimize a layout for a persona, ranging from the choice of the panels to be included (excluding information is important in reducing perceived complexity) and allocating screen real-estate to panels. Real-estate allocation techniques include using large zones with a single panel for critical information and small zones with many (tabbed) panels for less frequently needed information.

Item layouts

An item layout displays specific items and can also contain panels that display various aspects of an item. You can specify whether to enable users to create an item directly from

a layout when you configure it. The Create functionality can be available in addition to or instead of the Create function on a home page layout.

Note: You must create an item layout that includes at least a form panel to make the layout visible when opening an entity instance in Process Experience.

To create an item layout:

1. In **Workspace Documents**, open the entity.
 2. Click **Add > Layout**.
 3. In the Layout Details pane, define the following options for the layout.
 - Display Name - Provide a name that will identify the layout to users.
 - Name - Provide a name for the layout.
 - Description - Enter a description of the layout, such as information about the users for whom it is intended.
 - Preview Layout – Select this option to display the layout in a Preview panel.
 - Full Layout – Select this option to display the layout in a separate page.
 - Use to create new items - Select this option to enable users to create new items from the layout. See [Creating items using an item layout](#)
 - Item Url - If you want to create a web application that presents items to users, you can use this Url to present an item using a specific layout. For example, an Employee entity could be linked to an employee portal web site to display a layout that presents the employee's vacation balance information.
 4. Click **Add**.
- Tip:** If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.
5. Click **Configure**.
 6. Populate the layout with the required panels. See [Layout panels](#) and [Adding panels to a layout](#).
 7. Click  (Save).

Creating items using an item layout

An option called **Use to create new items** is available in the item layout design properties for an entity. When this option is selected, the layout provides a Create option in both the Create menu and in the Create panel if the application user has permission to create items.

The feature enables more complex user interfaces when creating new items. For example, when a layout includes a form, you can specify a Contents panel, Discussions panel, or action bar. The following options are available when a user creates an item within a layout:

- **Save** saves the item and displays it for the user to continue working.
- **Save and create another** saves the current item and provides a form to create a new item.
- **Cancel** discards the item and deletes the draft.

The following panels support presenting data during creation:

- Form
- Discussion
- Content
- Preview
- Actions

Other panels are not restricted or prevented from being added to the layout but the data will be shown only after the item creation is completed. For example the lifecycle panels, tasks, and progress can be used and will display the contents after the item has been created.

Home page layouts

A home page layout (application layout) is added to a project to provide users with a landing page. Users can freely switch between any application layout they have permission to use. Application layouts are launched with no item and can only include panels that do not require an item.

You can show or hide the Create option in a home page layout. For example, you might want to disable users from creating items on a home page that is designed as a dashboard.

To create a home page layout:

1. In **Workspace Documents**, select your project.
 2. Click  (New).
 3. Select **Other**.
 4. Search for **Home Page Layout** and select it from the results.
 5. In **Home Page Layout Details**, define the following options for the layout:
 - Display Name - Provide a name that will identify the layout to users.
 - Name - Provide a name for the layout.
 - Layout Type - Select **Home Page** if the layout will be used as a home page or select **Layout Panel** if the layout will be included in a Layout panel. Only layouts that are added with a type of Layout Panel can be added to a Layout panel.
 - Header Image - If you want to provide a customized logo for the home page, click **Select**, choose the image, and then click **OK**.
- The images used must be added to a web library definition location within the project. See [Adding a customized logo to a home page](#).

- Hide item creation - If selected, application users cannot create items and the Create menu is not shown in the header area. If not selected, the Create menu is available for users to create items.
6. Click **Configure**.
 7. Populate the layout with the required panels. See [Layout panels](#) and [Adding panels to a layout](#).
 8. Click  (Save).

Layout panels

The system provides the following layout panels. The panels that are available depend on whether you are configuring a home page layout or an item layout.

- [Actions](#)
- [Activity flow summary](#)
- [Breadcrumbs](#)
- [Business Workspace](#)
- [Contents](#)
- [Create](#)
- [Discussions](#)
- [Emails](#)
- [External UI](#)
- [Form](#)
- [iHub](#)
- [Layout](#)
- [Lifecycle Progress](#)
- [Lists](#)
- [Preview](#)
- [Results](#)
- [Tasks](#)
- [Web Content](#)
- [Work](#)
- [XForm](#)

The following sections describe each panel.

Actions

Use the Actions panel to present the actions associated with an item as a set of buttons and menus. An application user clicks a button to invoke an action. If the action requires parameters, it prompts the user to enter the parameters and passes them as part of invoking the action. This panel is used in item processing pages and is incorporated as a

subpanel inside the results panel to perform actions on selected result items. The Actions panel is available only for item layouts. When you add an Actions panel, you select the action bar to be shown for it. See [Creating an action bar](#).

Activity flow summary

Use the Activity flow summary panel to select the activity flow tasks required to process an item such as a claim, order, or service request in your application so that an application user can view the activity flow properties for different action performed on the activity flow tasks. It is available only when the entity contains an [Activity flow](#) building block.

You can select any of the following properties to show as columns in the activity flow list in the Task panel in your application.

Property	Description
Acted by	User who most recently worked on the activity flow.
Acted on	Date when the activity flow was most recently worked on.
Category	Category of the activity flow.
Completed on	Date when the activity flow was completed.
Description	Description of the activity flow.
Display Name	Name that is displayed to users of the activity flow.
Initiated on	Date when the activity flow was initiated.
Progress	Progress of the activity flow.
Started by	User who started the activity flow.
Status	Status of the activity flow.

The Display Name, Initiated on, Progress, and Status properties are selected by default but you can select other properties as needed. The Display Name property is required and is disabled so that you cannot clear it.

The Task list presents tasks from the activity flow model. When an application user clicks an initiated activity flow, the activity flow details are displayed in the Activity flow summary layout configured in the activity flow model.

Breadcrumbs

Use the Breadcrumbs panel to provide a trail back up to the home page where a user started navigation. This functionality is provided as a panel so that you can choose whether to devote screen real-estate for this capability and to decide where on the page it should go.

When creating an item while viewing another item layout, the breadcrumb on the new item layout displays the previous item viewed in the breadcrumb trail.

The Breadcrumbs panel displays the current item that the application user is displaying. It also enables the user to navigate back through the route to the item the user is currently

viewing. In the following scenario, the Breadcrumbs panel displays the current item and also the item being viewed when the **Create and open** option was selected.

- Open an existing item from a Results panel.
- While viewing the item click the **Create** option.
- When creating the new item (which can be from any project available), select **Create and open**.

The breadcrumb in the new item page is displayed and enables the user to navigate back to the previous item viewed.

Business Workspace

Use the Business Workspace panel to display the folder browser widget that the Content Server exposes. Using the folder browser widget Users of your application can perform all the document management operations that are supported by Content Server. See [Adding a business workspace](#).

Contents

Use the Contents panel to display the content list in your application. It is recommended that you also include a Form panel so that users can see the item, its attachments, and a preview of a selected item. If the entity contains a To One or Child to Parent relationship, the configuration panel displays a Relationship Properties list from which you can select the related entity whose Contents panel will be shown to users of your application. See [Adding content](#).

Note: The Contents panel is not shown in mobile applications.

Create

Use the Create panel to display a list of all items that can be created. A user can select an item to create and can also filter the list. It is typically included on a home page and is an alternative to using the Create icon. Unlike the Create icon, the Create panel does not separately list Recent Items.

Discussions

Use the Discussions panel to provide a panel for the message board created by the Discussion building block. Typically, you would also include a form (such as the Default form) so that you can refer to the item information in the discussion.

If the entity contains a To One or Child to Parent relationship, the configuration panel displays a Relationship Properties list from which you can select the related entity whose Discussions panel will be shown to users of your application.

See [Adding a message board](#).

Emails

Use the Emails panel to provide email functionality in your applications. This enables users of your application to send, receive, reply to, and forward emails (and attachments) from a

selected item. This panel is not available for a home page layout. See [Providing email functionality](#) and [Creating an email template](#).

External UI

Use the External UI panel to display data from an external repository such as the Process Platform Inbox or Process Component Library. See [Predefined EIS connectors](#).

Form

Use the Form panel to display the properties of an item, usually for user input. First, you create forms as building blocks in entities. In Panel Properties, the Name option must be set to the name of the form to be displayed. See [Adding forms](#).

If the entity contains a To One or Child to Parent relationship, the configuration panel displays a Relationship Properties list from which you can select the related entity and form to be shown to users of your application.

iHub

Use the iHub panel to display a list of iHub reports and dashboards from which you can select a report to be displayed on a layout. See [Designing iHub reports on entity data](#).

To connect to the right iHub instance, Process Platform stores the iHub URL using the iHub Connection Manager.

Layout

Use the Layout panel to include other layouts within a layout. Only layouts that were added to the project with a type of Layout Panel can be added to a Layout panel.

Lifecycle Progress

Use the Lifecycle Progress panel to provide a high level view of an item's progress and so that an application user can view the completed, current, and future states of an item. If users deviate from a primary path, they can also view the alternate path from the deviation point. To use this feature, you must configure one transition for each lifecycle state as a primary transition. See [Lifecycle Progress panel](#).

Lists

Use the Lists panel to display the set of lists that the logged on user has permission to use. This panel assumes that the layout also includes a Results panel that displays the results of the selected list. It also assumes that the Results panel has the **Link to list panel** option enabled. On the Lists panel, the user can reorganize the set of lists, create new sections, drag lists between sections, hide lists, and rename lists. See [Adding lists](#).

Preview

Use the Preview panel to enable application users to preview an item inside a home page or item. The preview panel responds to the selection of an item in another panel, presenting

that item's entity preview layout. For example, a preview panel in an application layout that also contains a Results panel displays any item selected in the Results panel. The Preview panel can display many items at the same time, organizing them using tabs.

The Preview panel is integrated with the Brava viewer. If Brava is configured in Process Platform, the preview is shown using the Brava viewer. Otherwise, the default viewer is used to display the content.

Note: Files uploaded in an installation earlier than 16.0.1 cannot make use of the Brava viewer integration functionality. To use the Brava viewer with these files, install release 16.0.1 or later and configure Brava and then re-upload the files.

Results

Use the Results panel to display a list of items. It provides the main functionality of the standard user home page. When you add the Results panel to a layout you can choose to have the panel display the results of a fixed, specified list or to respond to the selection of a Lists panel elsewhere in the layout. Results panels are often used in item layouts embedded in tab containers tied to predefined lists.

The following options are available to a user in the Results panel:

- Pull out the filter panel from the left edge of the panel to specify filters to reduce the number of items presented in the results.
- See the current set of filters above the results.
- Use a menu to access functions to customize the columns presented in the results.
- Use the actions bar to invoke actions on items selected in the results.
- Use Pagination controls to page through large result sets.

The Results panel provides extensive personalization features. When a user returns to a layout page that includes a Results panel that is not tied to a specific list, it automatically displays the results for the last list used. The Results panel remembers how the user last configured each list's columns and the filters they used. The Results panel also attempts to remember which items the user most recently selected.

If the list is defined with a limit to the number of items to be displayed and there are more items available than can be displayed, the footer in the Results panel provides the ability to access multiple "pages" of result items. The following options are provided in the footer.

- Results per page - Enables the user to choose the number of results they wish to display in each page (up to the limit specified by the underlying object).
- Item range - Indicates the index of the items displayed in the current result page. For example, 1 - 20 indicates that the first 20 available items are being displayed.
- Total count - Indicates the total available number of items. This is only included if the view is configured to display this total.
- First - Displays the first page of result items.
- Previous - Displays the previous page of result items.
- Page - Indicates which page of result items is being displayed. This can be changed to skip to any page. Entering a number less than 1 displays page 1. If the total count is not

displayed, a user can enter a number that is past the last available page, in which case a blank result page is displayed.

- Next - Displays the next page of result items.
- Last - Displays the last available page of result items.

Tasks

Use the Tasks panel to select that tasks that must be performed to process an item such as a claim, order, or service request in your application. It is available only when the Lifecycle building block is added to the entity.

You can select any of the following properties to be shown as columns in the task list in your application.

Property	Description
Status	The current state of the task.
Subject	The subject of the task.
Assignee	The user or role that is responsible for the task.
Priority	The priority of the task. It is set between 0 to 5 (low to high).
Due date	The date when the task is due.
Delivery date	The date on which the task is created.
Start date	The date when the task was started.

The Status, Subject, Assignee, and DueDate properties are selected by default and you can select other properties as needed. The Subject property is required and is disabled so that it cannot be cleared.

The Task list presents tasks from the Lifecycle model and tasks from the sub-process that is triggered from the Lifecycle or on performing an action configured using a rule. When a user of your application clicks a task item, the task details are displayed in the task layout configured in Lifecycle model. When a user clicks a sub-process task, the classic Inbox View is displayed.

Note: When the lifecycle task due date elapses, the color of the task due date field in the task panel does not change until the page is refreshed.

Web Content

Use the Web Content panel to embed a web page in a layout. The design options for this panel specify the URL of the web page to be displayed.

Note: The Web Content Panel uses an inline frame (iFrame) to display the content. An IFrame is an HTML document embedded inside another HTML document on a website. Not all sites allow their pages to be hosted inside an iFrame. Some sites display a message stating that they do not allow it. Some simply not display properly in an iFrame.

You can specify an exact URL to be shown when the panel opens. For example, if a user's task is to look for newspaper articles that contain certain information, you might configure the URL as <http://www.nytimes.com/> to go directly to that page.

Alternatively, you can specify the URL using variable substitution syntax that enables you to include properties in the URL. These can be any of the system configuration properties. This is typically used to specify the host name in the URL so it can be changed without having to change the project's definition.

In an item layout, you can use any of the entity's properties to embed web pages from existing applications that take parameters in their URLs. For example, assume that an existing customer management application takes a URL with a CustomerId parameter to return a web page with customer information. You can include a Web Link panel in the layout for a loan application that specifies a URL of <http://MyCustomers.com/CustomerDetails?Id={item.Properties.CustomerId}>, using the CustomerId property from the loan item entity to provide parameters for the URL.

When the Web Content panel appears on an item layout, all of the properties of the current item are available to be used to generate a customized URL that is specific to the entity being viewed. You do this by using variable substitution that includes values from the entity type in the URL. For example,

<https://maps.google.com/maps?q={Item.Properties.ZipCode}&output=embed>

Note: The Item.Properties is case sensitive and must be typed as shown.

This dynamically constructed URL displays a map of the zip code that appears in the ZipCode property of the object type's property group (named Properties).

All such variable name references are case insensitive, but other parts of the URL and the values substituted for the named variables are as they were entered. The way in which the values are interpreted by the destination website is up to that site.

While the variables used in the URL would typically be properties in an entity (as in the previous example), any property in an entity can be used in the construction of the URL. Examples of the types of properties that can be used are:

- Item property - {item.<Element name>.<Property name>}
- Item ID - {item.id}.
- Current user's name and user id - You can access the user's user id by specifying user.Properties.UserId or just user.UserId and you can access the user's full display Name by specifying user.Properties.FullName or just user.FullName. Variables referencing user properties are not limited to use in panels that appear on object layouts. They can be equally useful on home page layouts.

The following user properties are available:

- {user.Properties.FullName}
- {user.Properties.Name}
- {user.Properties.PreferredLocale}
- {user.Properties.UserGuid}

- {user.Properties.UserId}
- {user.Properties.memberid}

Case matters with the property name of the user node. These all work:

- {user.Properties.UserId}
- {USER.PROPERTIES.UserId}
- {user.PROPERTIES.UserId}

But this does not:

- {user.Properties.USERId} (Notice the case is wrong USERId)

- System properties – system level properties that are available for use in applications. Typically, the value of these properties depends on the environment where the application is installed. The following system properties are available:
 - {system.baseURL} is the beginning URL portion for references to other OpenText applications running on the same server in the same organization.
 - {system.organization} is a reference to the current user's organization.

Case is not important with the system properties. All of these combinations work:

- {system.baseurl}
- [SYSTEM.baseurl]
- {system.BASEURL}

If a variable cannot be resolved (if it was mistyped, is unavailable, or does not exist), the value is not substituted and the variable reference is displayed as entered (enclosed in {}).

If the property value is empty, the parameter is substituted with null. Assume that a user creates items using the Appraisal object type. Each time the value of the city property changes, the Web Content panel displays a map of that city. The following example shows how this is specified:

<https://maps.google.com/maps?q={item.properties.CityP}&output=embed>

If the destination website does not provide meaningful feedback on malformed URLs, there are URL echo sites that can help to determine the resulting URL was by echoing it back to be displayed in the panel.

The capability to generate a customized URL that is specific to the object depends on the external system the user is connected to.

Note: The variables in the curly braces are limited to local properties - relationships are not supported. This is an issue for lifecycle layouts because the properties are not on the task, but on its parent, the case.

Work

Use the Work panel to display a summary of the lists that the application user has permission to use.

XForm

Use the XForm panel to provide two options for displaying an XForm within a layout. You can select an XForm that is available in the workspace using a drop-down list, or provide the URL to an existing XForm.

When selecting an XForm from the drop-down list, you can also provide optional parameters using the Data property. These parameters use the substitution syntax that enables the inclusion of entity properties within the xForm URL parameters. You can use any of the entity's properties so that the XForm displayed can use this data. For example, an asset management application has XForms that take a parameter to display a specific configuration of the asset infrastructure. It is possible to provide the AssetId as a parameter so the XForm provides the asset configuration information within the item layout. The Data property would be specified as follows:

```
AssetId={item.PropertiesAssetId}
```

To add more than one parameter, add an ampersand between parameters as follows:

```
AssetId={item.PropertiesAssetId}&InvoiceId={item.PropertiesInvoiceId}
```

While the variables used in the Data property would typically be properties in an entity (as in the previous example), other properties can be used in the construction of the xForm URL parameters.

The full list of other properties that can be used follows:

- Current user name and user ID - You can access the user's ID by specifying {user.Properties.UserId} and you can access the user's full name by specifying {user.Properties.FullName}.
- Solution variables – {config.variable} where 'variable' is expected to be defined as a solution variable within the containing solution using the Process Experience Administration tool.
- System properties – System level properties that are available for use in applications. Typically, the value of these properties depends on the environment where the application is installed. The following system properties are available:
 - {system.baseURL} is the beginning URL portion for references to other OpenText applications running on the same server in the same organization.
 - {system.organization} is a reference to the current user's organization.

Adding panels to a layout

You add panels to a layout by dragging them from the Panels area to the Desktop area. The decorations around sections and panels is called the chrome. Panels do not display any chrome. Margins, borders, title bars, and so forth are added around the panels as configured in the containers that organize the panels. For example, you can choose to stack two panels in a section with no borders between them so that they will appear to a user as if they are a single multi-tabbed panel. The chrome in a tab section is not configurable.

Layouts subdivide screen real estate to present information. However, the amount of space available can vary widely when the same layout is accessed from a desktop with a large monitor or on a mobile phone with a tiny screen and all the intermediate sizes. To simplify this, responsive design techniques automatically rearrange the panels in a layout based on the available screen real estate. You design a layout for the largest sized screen and the panels are automatically resized for smaller screens. In addition to rearranging the panels, each individual panel responds to reduced screen real estate within its panel. For example, the Lists panel can automatically shift from a grid presentation of results to a "paragraph" presentation that works better in a small space. There is no general mechanism for these panel specific behaviors - they are designed specifically for each panel.

Note: At run time, the first and second panels in a layout are checked to determine whether they are Action or Breadcrumbs panels and whether they are defined at full width. If they are, they are both merged into the Process Experience header. However, if the uppermost panel is an Actions panel, the second is a Form panel, and the third is a Breadcrumbs panel, the Actions panel is merged into the header and the Form and Breadcrumbs panels are displayed normally.

To add panels to a layout:

1. In **Workspace Documents**, select an item layout and click **Configure**.
For a home page layout, you add panels when you create the layout.
2. Select a panel and drag it to the left, right, top, or bottom border of a page or panel on the layout canvas.
A highlighted border indicates where the panel will be dropped.
Other panels that are already included in the layout are automatically resized to accommodate the new panel.
To create a panel as a tab, drag it to the middle of the destination panel. As you select a panel, its general properties are displayed in the General area.
3. In **Name**, type the name to be displayed to users. The name should not exceed 128 characters.
4. In **Element Name**, type the name of the panel.
5. In **Description**, type a description of the panel. The description should not exceed 1024 characters.
6. In the **Chrome** list select Full, Container only, Title only, or None to specify how the panel is displayed in your application. For a tabbed panel, chrome is always displayed.
 - If **Full** is selected, the panel contains a header, borders, and white background. See [Defining a custom icon for a panel header](#) for information about how to customize the icon for Full chrome.
 - If **Container only** is selected, only the container is shown.
 - If **Title only** is selected, the panel provides a header chrome but no icon is displayed.
 - If **None** is selected the panel has no header or borders and the background is transparent.

1. If your layout contains any of the following panels, configure the properties specific to the type of panel in the Panel Properties area.
 - **Actions** - Select the action bar to display in the panel.
 - **Contents** - Select the content to display in the panel. The content can be from the entity you are configuring or from an entity with which it has a To One or Child to Parent relationship.
 - **Discussions** - Select the discussion to display in the panel. The discussion can be from the entity you are configuring or from an entity with which it has a To One or Child to Parent relationship.
 - **Form** - Select the form to display in the panel. The form can be from the entity being configured or from an entity with which it has a To One or Child to Parent relationship.
 - **iHub** - Select the report to display in the panel.
 - **Layout** - Select the layout to include in the panel. Only layouts that were added to the project with a type of Layout Panel can be added to a Layout panel.
 - **Results** - Select one of the following options:

Link to list panel specifies that the items selected in the Lists panel should be displayed in the Results panel. Only one Lists panel should be added to the layout for this to work.

Run list displays the results for the list that is selected from the drop-down list. All lists in the solution are available for selection. However, avoid selecting a list if the users who will open the layout have not been given access to it.

- **Web Content** - Enter the URL to be displayed in the panel.
You can enter a simple static URL such as
<https://maps.google.com/maps?q=03062&output=embed>.
You can also use variable substitution to generate a customized URL that is specific to the object being viewed. See the description of the Web Content panel later in this topic.
- **XForm** - Select XForms or XForms URL and enter the data or URL to display in the layout.

2. Click  (Save).

Adding panels from a related entity to a layout

If an entity contains a To One or Child to Parent relationship (see [Adding relationships](#)), you can include any of the following panels from the related entity in your layout:

- Form - Display a Form panel from a Case Management entity within a task layout.
- Contents - Enable access to the Contents panel from within all child layouts without having to access the parent layout.
- Discussion - Display a shared Discussion panel between a Release Management entity and all of the related Change Management entities.

If the entity for which you are configuring a layout has a To One or Child to Parent relationship, a Relationship Properties list is available in the Configuration panel.

- To include a related form, you select the related entity and form to include.
- To include a Content panel, you select the related entity.
- To include a Discussion panel, you select the related entity.

You can include multiple panels from related entities in a layout.

Including multiple forms in a layout

Many applications need to make a large amount of information available at the same time. One way to organize this information and keep the user interface simple is to include different types of information on different forms and then make each form accessible as a tab on the same page in Process Experience. For example, when a clinician selects a patient from a list, tabs can be provided to view the patient's medical history, drug list, interview questionnaire, and upcoming appointments all from a single location.

To include multiple forms as tabs in a layout:

1. Create a separate form for each type of information you want to make available.
2. Create a layout where the forms will be accessed.
3. Drag a form panel to the layout.
4. Drag additional form panels to the layout, dropping each one in the center of another form panel.
5. Select each form panel and select the form to display along with its other properties.
6. Save the layout.

See [Adding layouts](#).

Home page example

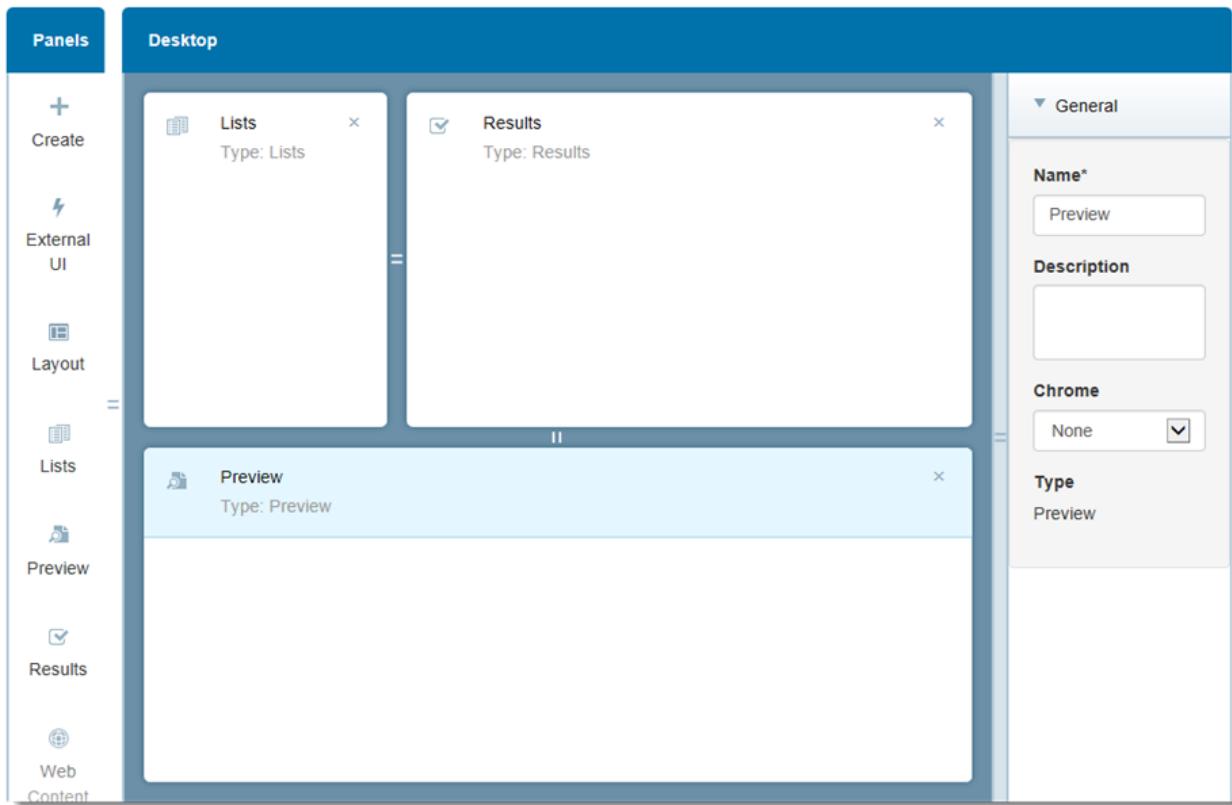
Generally, a basic home page includes the following panels:

- A Lists panel to display the user's lists.
- A Results panel to display the results of a selected list.
- A Preview panel so that users can preview items before opening them.

To create a home page that contains a Lists, Results, and Preview panel:

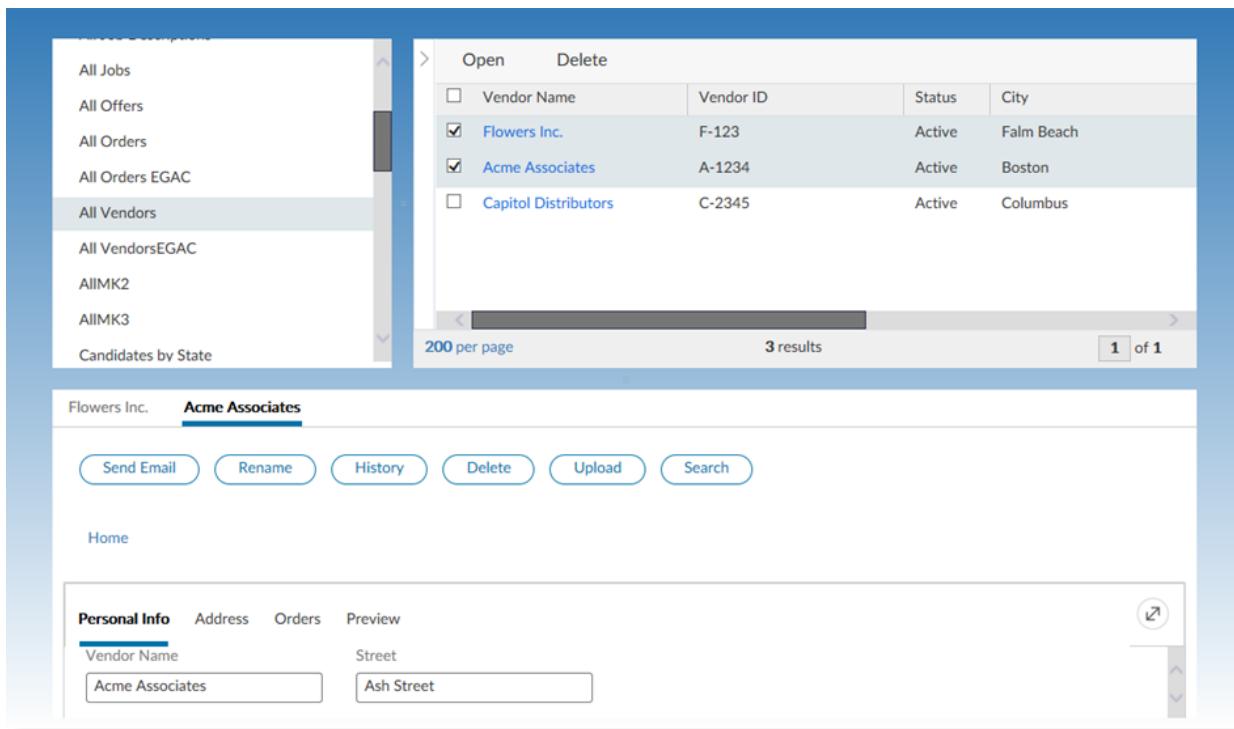
1. Create lists that will present items in a list of results and, if necessary, filter the results of the lists. See [Adding lists](#).
2. Create a form that will be used to display a preview of the items in the Results panel. See [Adding forms](#).

3. Create a Preview layout.
 - In Chrome, select **None**.
 - In **Panel Properties > Form** select the form to be used to display the preview.
 - Optionally include an Action panel in the layout.
4. Create a home page layout that contains a Lists panel, a Results panel, and a Preview panel.



In Process Experience, this appears as follows.

- When a user selects an item in the Results panel, a preview is displayed in the Preview panel (no chrome is displayed) using the specified form.
- If multiple items are selected in the Results list, each item is shown as a separate tab.
- If the Preview layout for the entity includes an Actions panel, the user can perform the specified actions on the item.



Adding a customized logo to a home page

You can replace the standard Process Experience logo with a customized logo. For example, a home page for an HR application can include that department's logo. The images available for selection are taken from the CWS project where the home page resides.

Tip: In addition to changing the logo in the file system, the logo image can be customized by creating a theme. See [Creating a theme](#)

Before you can add an image to a layout, you need to include it in your project or in a folder in your project. For example, you can add the images to a project folder called Images.

To make the images available in Process Experience, you must add a Web Library Definition to your project to include your image folder before you publish the project. If you later need to specify a different image folder, you must re-publish the project.

To create a folder for images (optional):

1. Right-click the project and select **New > Folder**.
2. Replace **Untitled Folder** with the name you want.

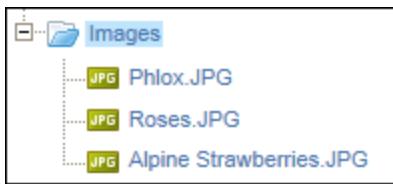
To include images in a project or folder:

1. Right-click the project or folder and select **Upload Document**. The Upload Document wizard begins.
2. Click **Browse** and select the image file.

Note: The image browse dialog box may not recognize certain file types as images. For example, .ico and .svg files are not supported.

3. If necessary, modify the displayed description.
4. Click **Finish**.

In the following example, three images were added to the Images folder.



To add a Web Library Definition to your project:

1. Open your workspace, if it is not already open.
2. Click (New).
3. Select **Other**.
4. Type **Web Library Definition** in the search box, and then select it in the search results.
5. In Web Content Settings, browse to the location of the image folder and click **Save**.
6. Click **OK**.

To include an image in a home page layout:

1. Open the home page layout.
The Home Page Layout Details window opens.
2. Click **Select**, choose the image, and then click **OK**.

To remove an image from a home page layout:

- Click **Remove** in the Home Page Layout properties.

To select a different image:

- Click **Upload** in the home page layout.

Adding forms

Typically, users view or enter information using a form. For example, a form can present an entity's properties so that a user can create items. Forms can organize information into various types of containers. They can also contain images and their appearance can be customized using formatting options. You can create as many forms as your business requires.

Note: Process Experience users can print instances of the forms that you design. For example, a user can print a form that provides information about a particular order. See the

Process Experience online help or the *Process Experience User's Guide* at [Open Text My Support](#).

For each entity, you must create a form named Create. This form is used when a user creates a new item in Process Experience. For example, if Create forms are created for the Vendor entity and the Invoice entity, options called Invoice and Vendor are available from a menu when a Process Experience user clicks **Create**.

If the user selects **Vendor**, the Create form for the Vendor entity is displayed. If no Create form is configured for an entity, items based on that entity are not available in the list of items that can be created in Process Experience, either from the Create menu or from the Create panel in a layout.

Important: The name of the Create form must begin with an uppercase letter. If the form is named create, a blank form is displayed in Process Experience.

In addition to designing Create forms, You can also design other forms that are displayed when a Process Experience user opens an item. There are two ways to do this:

- Specify the form to use when you add a Form panel to a layout. For example, you can create a form named ViewVendor for items that are opened from a list.
- If you do not create a layout that specifies a specific form to display, you must create a form named Default for the system to use to display items when they are opened.

If you do not create a layout that specifies another form, and you do not create a form named Default, Process Experience users will get an error trying to view an item, because the system cannot determine what form to use.

Process Experience users who have Create permission to create items must also have Update permission to enable them to update the properties in the item.

Note: The functionality available to forms created for EIS entities aligns with the capabilities of the EIS connector that enables those entities. For instance, you may be able to create a form for EIS entities such as those provided by the Case360, MBPM, PCL, or Inbox connections. However, if the connector does not allow create, update, or delete access, the form allows only read access of the external data.

Important: When [publishing an entity](#), components on forms are validated. Most validation errors are due to missing configuration information. In these cases, you can open the form, add the missing configuration, and repeat the publishing process. In very rare situations, form validation errors are caused by orphaned form components. Since orphaned form components are not visible on the form canvas for you to remove, the entire form must be removed and recreated.

Creating a form

To create a form, you first add the Form building block to the entity and then you add properties and other components to it and customize them based on your requirements. You use the properties pane on the right side of the window to configure each component. You can also organize the properties and other components in various ways. See [Positioning components on a form](#).

When you create a form, the Create URL property is automatically shown on the form properties with a supplied value. Each Create form has its own specific URL. You can copy this URL and share it with customers or link to it from external applications outside of Process Experience (for example, a third party HR application).

This enables customers or users of the external systems to directly access a specific Create form to add new employees to the system. When they do this, the Create form contains all the properties (first name, last name, social security number, date of birth, and so forth) required to add an employee to the system.

Although you cannot change the supplied URL, you can append a query string token of "rurl" with a relative path to redirect it.

The following example shows a redirected Create URL. The redirect relative URL is used only after the user clicks **Create** or **Cancel**.

```
http://alqb-tnw7x64vm.lab.opentext.com/home/system/app/processExperience/web/
perform/create/00505601050711E6FAD6E59DA5E516C1?rurl=
home/system/app/processExperience/web/perform
```

The following components are available for configuring a form.

Component	Description
Property	Defines the information that users will enter or see when they open the form. For example, if you are designing a form that will be used to create a master list of vendors, you might include properties such as Name, Address, and other descriptive information. All of the properties that were defined for the entity or for any related entity are available for you to include.
Relationship	Displays a list of related entities and their properties.
Actions	Places action buttons directly on the form instead of in the Action Bar.
Container	Enables grouping of other components.
Tab Container	Presents a set of containers as tabs that a user can switch between.
Horizontal Line	Displays a horizontal line in the layout that can be used to visually separate portions of a form.
Image	Includes selected images and color in forms.
Text	Adds specified text to a form.
Form	Displays a list of other forms that were created for the entity and its related entities so that you can nest one form within another.
Hyperlink	Adds a hyperlink to redirect to another page.

A form opens with a default width and height that are shown in the Form configuration panel. When you add, select, or resize a component, the values are updated with the width

and height of the selected component. These values are read only. You can resize a component using the pointing device but you cannot resize it by editing the displayed values.

To add and configure a form:

1. In **Workspace Documents**, open the entity.
 2. Click **Add > Form**.
 3. In the Form Details pane, type the Display Name, Name, and Description of the form. The value in **Form URL** is automatically supplied and cannot be changed.
 4. Click Add.
- Tip:** If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.
5. Click **Configure**.
 6. Configure the form as described in the following topics:
 - [Adding properties to a form](#)
 - [Adding relationships to a form](#)
 - [Adding containers to a form](#)
 - [Adding hyperlinks to a form](#)
 - [Adding actions to a form](#)
 - [Adding images and color to a form](#)
 - [Nesting forms](#)
 - [Disabling or hiding information on a form](#)
 - [Positioning components on a form](#)
 - [Keyboard shortcuts for forms](#)
 - [Tracking form progress](#)
 - [Drop list series example](#)
 7. Click  (Save).

Tip: You can organize various types of information and keep the user interface simple by including each type of information on a separate form and then making each form accessible as a tab on the same page in Process Experience. See [Including multiple forms in a layout](#).

Adding properties to a form

A form can include properties from the entity you are configuring or from a related entity. For example, if you are configuring an order entry form, you can include properties such as Order Number, Item, Vendor, and so forth. Properties can be displayed directly on a form or within a container.

If the entity contains a Title building block, a Title property is available for adding to the form. The Title building block specifies whether the title will be text that is entered by the user or whether it will be a read-only title that is supplied by the system. See [Adding a title](#).

You specify how each property that you include on a form will be presented to a Process Experience user. The following list describes each presentation type.

The presentation options that are available depend on the type of property.

- Multi-line - Displays a value as a multi-line text box. Scroll bars are available for moving up and down within the text box.
- Text box - Displays the value in a text box.
- Text only - Displays the value as a label.
- Progress bar - Displays the value as a progress bar when the user opens an item. This option is available for numeric properties only. See [Tracking form progress](#).
- Drop List - Displays values in a drop-down list from which a user makes a selection. A user of your application can also choose to show an icon in place of, or along with, the value. This presentation option is available for a drop list for Boolean, static Enumerated Text, and static Enumerated Integer property types. See [Drop list series example](#) for the drop-list presentation.
- Radio Button - Displays values as radio buttons from which a user makes a selection. Be sure to allow enough room so that the list does not overlay other properties on the form. You can use the **Distribute into column(s)** option to display the values in columns.
- List Box - Displays values in a list from which a user makes a selection.
- Duration - Enables users to enter or edit duration values as numbers of units and an adjacent label shows the result.
- Rich text - Displays the value in the rich text editor. This presentation is applicable for Long Text properties. The value can be formatted with the following options:
 - Style Formatting (Bold, Italic, Underline, Strikethrough, Remove format, Font name, Font size, Text color, Background color).
 - Paragraph Formatting (Insert/Remove Numbers list, Insert/Remove Bulleted list, Decrease Indent, Increase Indent, Block quote, Insert Horizontal Line).
 - Undo and Redo.

Content formatted with the Rich text presentation will have HTML tags when shown in other presentation types and in List columns.

Note: If you copy (map) a long text property containing rich text, be sure that the content is handled as text.

- Image – Enables users to upload and view an image in your application. This presentation is applicable for property type Image. You can set border styles such as Style, Width, Color, and Border Radius.

Note: The following section explains how to add properties from the same entity to which you are adding the form. See [Adding relationships to a form](#) for information about how to add properties from a related entity.

To add properties to a form:

1. In the Components pane, expand the list of properties and drag the properties you want to the Presentation pane.

Tip: You can drag the properties individually or you can drag all of them at the same time by dragging the Properties heading.

2. In **Presentation**, select a presentation option for each property.
3. If necessary, select any of the following options:
 - Click **Default text style** and select a text font, size, color, and style.
 - Select **Required** if a Process Experience user must enter a value for the property. Select **Read Only** if a Process Experience user cannot change the value for the property.
 - Select or clear **Show Label** to specify whether to show a label for the property in Process Experience. This option is selected by default. The name of the property is supplied but you can type a different name in the text box. You can also click **Default label style** to specify the text font, size, color, and style for the label.
 - Select the position and alignment for the label. For **Top** position **Left**, **Center**, and **Right** alignment options are available. For **Left** position **Left** and **Right** alignment options are available. For better alignment of multiple controls use the icons in the Align toolbar. See [Designing, testing, and delivering applications](#) for a description of each icon.
 - Select or clear **Show on mobile devices** to indicate whether to display the property on mobile devices. This option is selected by default.
 - For Currency property types, a **Default currency format** option is available. When this option is selected (the default), the currency presentation format in your application is derived from user's locale. You can customize the currency format by clearing the option and then setting the following options for the currency presentation:
 - Positive values - Placement of the currency symbol for positive values.
 - Negative values - Placement of the currency symbol for negative values and specifies whether the negative sign is represented by parentheses or the negative sign.
 - Digit grouping - Number of integer digits that appear in a group.
 - Digit separator symbol - String that separates groups of integers.
 - Decimal separator - String that separates integer and decimal digits.
4. If you want to associate a property with a category that will be disabled or hidden in Process Experience, select it and then select or type a name in **Category**. See [Disabling or hiding information on a form](#).
5. Click  (Save).

Adding relationships to a form

By adding a relationship to a form, you can include properties from a related entity on the form and specify the available actions that are available for those properties in Process Experience. For instance, users may be able to browse for items and modify (permissions

allowing) their values. You select the list that will be available for browsing when you design the form. For example, when creating an order, users may need to browse through a list of vendors to select the vendor for the order. It is a good idea to add informational text to guide users. For example, you might add the following text next to the Browse icon:

Click Browse to select a vendor for the order.

When the user clicks **Browse**, a list of vendors is shown.

Following are some suggestions for defining and managing the lists that will be used with relationships in forms:

- Instead of selecting a list that is used in the Lists panel in Process Experience, it is best practice to create separate "internal" lists that include only the properties that are required for Process Experience users to select a related item from a form.
- To simplify working with the application, use a naming convention to differentiate the internal lists from the lists that are shown in the Lists panel.
- Clear the **visible to users** option on your internal lists.

Caution: In forms, an entity with a Deadline building block displays a relationship called DeadlineInstance in the Components panel. This is an internal relationship that should not be used when designing a form.

See the following sections:

- [Adding a To One relationship to a form](#)
- [Adding a To Many relationship to a form](#)
- [Defining filters on form relationship controls](#)

Adding a To One relationship to a form

In the Components pane, a To One relationship is prefaced with [0..1]. When you drag a To One relationship to a form, a Presentation option is provided so that you can specify how items should be presented when a user browses in Process Experience.

The following options are available:

- **Buttons (only)** - To select an item, a user selects the corresponding button (as for Text box). However, using this option, you can include additional details directly on the form. For example, when a user selects a vendor, information such as vendor ID, status, and address information can also be included on the form. Only the icons that you select are shown in the list of items. When a user selects an item, the specified properties are shown in the container.

Click the Browse icon to select a vendor   

Vendor Details	
Vendor Name	Street
Allied Associates	Andover Street
Vendor ID	City
A-111	Amherst
Status	State
<input checked="" type="radio"/> Active	AL
<input type="radio"/> Inactive	
<input type="radio"/> None	

- **Text box** - To select an item, a user selects the relevant button. The text box is updated when the selection is made.

Vendor Name	Vendor ID	Status
<input type="radio"/> Allied Associates	A-111	Active
<input type="radio"/> Best Business	B-222	Active
<input type="radio"/> Capitol Company	C-333	Active
<input type="radio"/> Danforth Distributors	D-444	Active

- **Drop list** - To select an item, a user drops down a list and makes a selection.

- Select -

A-111

B-222

C-333

D-444

P-123

E-567

M-678

- **List box** - To select an item, a user scrolls through a list.



- **Drop list series** - When you choose this presentation, the browse button is converted to a container and the properties that you choose from the selected worklist are created within the container. You cannot drag properties into or out of the container. The order of the properties in the Properties list is the same order in which the related properties are shown in the Components pane. A Duration property is not searchable so it is not added to a drop list series. The presentation of individual properties inside the container cannot be changed.

To select an item, a user makes a selection from each drop list in succession. After the user makes a selection from Make, Model becomes enabled. After selecting a Model, Year becomes enabled.

The screenshot shows a form with the following fields:

- CustomerName:** A text input field containing "John".
- Entity Picker Container:** A section containing three dropdown menus:
 - Make:** A dropdown menu showing "Toyota".
 - Model:** A dropdown menu showing "Camry".
 - Year:** A dropdown menu showing "- Select -" at the top, followed by "2009" (which is highlighted in blue) and "2010".

To add properties from a To One relationship to a form:

1. In the Components pane, drag the To One relationship to the Presentation pane.
2. In **Presentation**, select an option for presenting the property in Process Experience.
3. In **Browse list**, select the list to be made available when a user browses.

The **Dynamic loading** option specifies whether the list is loaded by default in Process Experience. This option is not selected by default. This option is available when **Button(s)** only or **Text only** is selected as the presentation type.

In Process Experience, dynamic loading behavior for the list works as follows:

- When the list is browsed, initially no results are shown.
- The Filter panel is shown by default.
- Results are retrieved only when the user enters the filter criteria.
- Unlike the normal lists, the personalized filter settings are not displayed the next time the list is browsed.

4. In **Related property**, select the property to be displayed.
This option is not available if Button(s) only is selected as the presentation.
 5. Click **Default text style** and select a font, size, color, and style for the property.
This option is not available if Button(s) only is selected as the presentation.
 6. In **Show Label**, specify whether to show a label for the property in Process Experience.
This option is selected by default. The name of the property is supplied but you can type a different name in the text box. You can also click **Default label style** to specify the text font, size, color, and style for the label.
 7. In **Actions**, select the options to be available for the property in Process Experience.
 - Browse - Enables the user to search for and select an item from the related entity. For a To One relationship, this option is selected by default and cannot be cleared.
 - Create - Enables the user to create an item.
 - Clear - Clears data from the field. The data is still available and can be browsed and selected again.
 8. Select or clear **Show on mobile devices** to indicate whether to display the property on mobile devices. This option is selected by default.
 9. If you want to associate the property with a category that will be disabled or hidden in Process Experience, select it and then type a name in the Category box in the right pane. See [Disabling or hiding information on a form](#).
- Note:** Children cannot be browsed or cleared, so Browse and Clear buttons are not available within a parent-child container. Not all options are available for all relationships. For instance, Create is not available when the relationship's target is a child entity from another relationship.
10. Click  (Save).

Tip: You can alternatively drag and position a related property and the relationship to the form individually and then configure each individually.

Important: In Process Experience, a related property on a form is disabled until the relationship is established by clicking **Lookup**. Therefore, because it is not possible to enter data for a disabled property, a required related property does not get a validation error if the user does not click **Lookup**. On a Create form, all required properties, including related properties, are validated when the user clicks **Create**. An error message is displayed and the Create action does not proceed if a value was not provided for any required property.

Adding a To Many relationship to a form

In the Components panel, a Peer-to-Peer To Many relationship is prefaced with [0..*]. A Parent-Child To Many relationship is prefaced with [1..*].

When you add a To Many relationship to a form, a special type of container called a repeating group container is automatically included on the form. When you include properties from the related entity in this container, Process Experience users can perform various functions on the related entity. For instance, a user may be able to open, clear, delete, create, or browse for an order.

Note: In the Create dialog box and the direct Create URL, a container for a To Many relationship does not provide the option to open an item, since navigating to the opened item would result in the work being done to create the source item to be lost.

Caution: In entity forms, the LifecycleTask relationship from the Lifecycle building block is read only. Performing actions such as Create and Delete on the LifecycleTask Repeating Group Container will lead to unexpected errors.

A repeating group container can be presented to Process Experience users in the following ways. After you save the form, you cannot change the presentation.

Also, for any repeating group container nested inside another repeating group container, the presentation is same as the outer repeating group container.

Repeating Group - Presents the related entities in a paragraph format, showing each related entity in a new group. Properties are displayed individually with or without their property labels.

The following example shows the Repeating Group presentation in Process Experience.

If you place a repeating group container in a section, you can have it automatically resize as items are added or removed in your application, you can close up any unused space or provide more space when users work with the form. See [Designing responsive forms](#).

Repeating Group Container

+

▼ 1

Id	Started Time
<input type="text" value="16,385"/>	<input type="text" value="10/02/2008 12:00 am"/>
Number	Employed
<input type="text" value="25"/>	<input type="radio"/> True
	<input checked="" type="radio"/> False
Name	<input type="radio"/> Unknown
<input type="text" value="Frank"/>	

▼ 2

Id	Started Time
<input type="text" value="16,390"/>	<input type="text" value="05/12/2015 12:00 am"/>
Number	Employed
<input type="text" value="25,221"/>	<input checked="" type="radio"/> True
	<input type="radio"/> False
Name	<input type="radio"/> Unknown
<input type="text" value="Chris"/>	

▶ 3

▶ 4

The information (for each instance) in a repeating group container can be expanded or collapsed. This is especially useful when a form contains nested repeating group containers. By expanding and collapsing the information in each container, Process Experience users can access to a large amount of information with minimal scrolling. When the information in a repeating group container is collapsed, a title is automatically provided to identify its content.

In Process Experience, the title is shown based on entity property value, in this order:

- If there is a Title building block in the target entity, the value of the entity title property is shown as the title.
- If there is no Title building block or the entity title value is empty, the value of the first required Text property found is shown as the title.
- If there is no required Text property, the value of the first required property that is not of type Duration or Boolean is displayed.
- If there is no required Text property or other type of required property (other than Duration or Boolean), the row number is displayed as the title by default.

In Entity Modeling the title is shown as <Title> and the first required property (preferably text) or row number is shown as a tooltip.

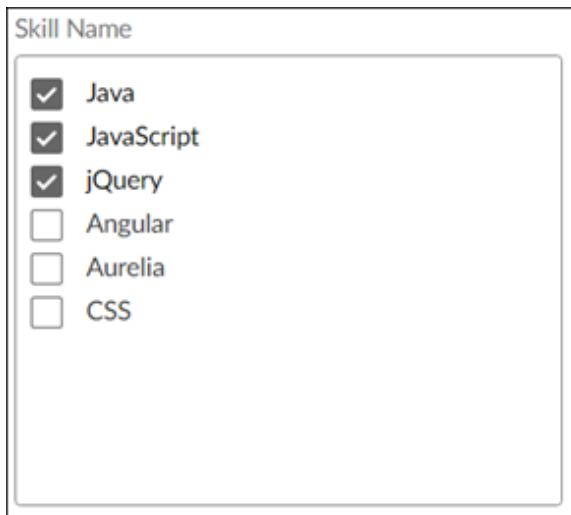
Grid - Presents the related entities in a spreadsheet format, with each related entity in a row its properties in columns. Using the Grid presentation, Process Experience users can add a new row, delete one or more rows, show or hide columns, sort data on columns, and filter data.

You can add a To One relationship to a grid column. Only the Text box presentation is supported for a To One relationship in a grid. In Process Experience, application users can select Browse, Create, and Clear actions from the drop-down menu of the grid cell. Once the related item is selected, the property value of the selected item is displayed in the grid cell.

Note: On mobile devices, the Grid presentation is shown as a Repeating Group presentation.

Note: The subgrid can no longer be added to a parent grid. Use the repeating group to include nested relationship content in a form.

Multiselect - Presents the related entity items in a multiselect check box format. It shows all the items from the relationship with related items selected. You can select the property to display to represent the relationship. The following example shows the Multiselect presentation in Process Experience. Users can link or unlink one or more items by selecting or clearing the check box, but they cannot create new items.



Note: On mobile devices, the Multiselect presentation will not be shown.

To add a To Many relationship to a form:

- Drag the To Many relationship to the Presentation pane.
A repeating group container is automatically added.

To define the attributes of a repeating group container:

1. In **Type**, select **Repeating Group**, **Grid**, or **Multiselect**.

Note: Only the repeating group type supports nesting. Multiselect and grid do not support nested relationships.

2. Select the repeating group container and, if necessary, specify the following attributes:
 - **Show container heading** - Specify whether to show a label for the container. The name of the entity is supplied by default but you can type a different name in the text box (available only for Repeating Group or Grid presentations).
 - **Default text style** - Select a text font, size, color, and style.
 - **Default label style** - Select a text font, size, color, and style.
 - **Default background** - Select a color and image. See [Adding images and color to a form](#).
 - **Default border** - Select the style, width, and color for the border of the container.
 - **Default page size** - Select this option to load 200 rows (the default) at a time in your application. To load a different number of rows at a time, clear the option and specify the number of rows in the text box that opens. When a user scrolls to the last row (200 rows for the default) the next 200 rows are loaded. This can improve data presentation if there are many rows to load.
 - **Show on mobile devices** - indicate whether to display the property on mobile devices.
3. In **Default row state**, select **Expanded** or **Collapsed** (available only for the Repeating Group presentation).
When **Expanded** is selected, the container is shown in Process Experience with all rows expanded. The default selection is **Collapsed**.
4. If you want the repeating group container to automatically re-size in your application, place it in a section and select **Auto-resize as items are added or removed**. It is optional to specify the number of items to display before providing a vertical scroll bar. When it is not specified, the repeating group container has no maximum height - it grows and shrinks based on the content and does not display a scroll bar. See [Designing responsive forms](#).
5. In **Actions**, select the options to be available on the form in Process Experience. All options are selected by default.
 - Open - Opens the selected item.
 - Clear - Clears the selected item.
 - Delete - Deletes the item.
 - Create - Enables the user to create an item.
Select **In a new row** to create an empty target item and then provide the values

inline in the repeating group or grid.

Select **In a dialog box** to open the Create form for the target entity and then provide the values in the Create form.

- Browse - Enables the user to search for and select an item from the related entity.

Note: Not all options are available for all relationships. For instance, Create is not available when the relationship's target is a child entity from another relationship. Clear and Browse are not available in a Parent Child relationship.

6. In **Browse list**, select the list to be made available when a user browses.

The **Dynamic loading** option specifies how the selected list is loaded when a Process Experience user browses. This option is not selected by default.

If this option is selected, the browsed list is shown as follows:

- Initially no results are retrieved and the filter panel is shown in expanded mode by default.
- Results are retrieved only when user enters the filter criteria.
- Unlike the normal lists, the personalized settings are not displayed the next time the list is browsed.

7. If you want to associate the container with a category that will be disabled or hidden in Process Experience, select it and then type a name in the Category box in the right pane. See [Disabling or hiding information on a form](#).

To include properties in the repeating group container:

1. In the Components pane, expand the list of properties for the related entity.
2. Drag a property to the repeating group container.
3. In **Presentation**, specify how the property will be displayed in Process Experience. The selections that are available are based on the property type.
 - Multi-line - Displays a value as a multi-line text box. Scroll bars are available for moving up and down within the text box.
 - Text box - Displays the value in a text box.
 - Text only - Displays the value as a label. This option is not available for [Dynamic enumerations](#).
 - Drop List - Displays enumerated values in a drop-down list from which a user makes a selection. A user of your application can also choose to show an icon in place of, or along with, the value. This option is available for Boolean, static Enumerated Text, and static Enumerated Integer property types.
 - Radio Button - Displays values as radio buttons from which a user makes a selection. Be sure to allow enough room so that the list does not overlay other properties on the form. You can use the **Distribute into column(s)** option to specify the number of columns for displaying the radio buttons. This option is not available for [Dynamic enumerations](#).

- List Box - Displays enumerated values in a list from which a user makes a selection. This option is not available for [Dynamic enumerations](#).
- Duration - Enables users to enter or edit duration values as numbers of units and an adjacent label shows the result.
- Rich text - Displays the value in the rich text editor. This presentation is applicable for Long Text properties. The value can be formatted with the following options:

Style Formatting (Bold, Italic, Underline, Strikethrough, Remove format, Font name, Font size, Text color, Background color).

Paragraph Formatting (Insert/Remove Numbers list, Insert/Remove Bulleted list, Decrease Indent, Increase Indent, Block quote, Insert Horizontal Line).

Undo and Redo.

Content formatted with the Rich text presentation will have HTML tags when shown in other presentation types and in List columns

- Image – Enables users to upload and view an image in your application. This presentation is applicable for property type Image. You can set border styles such as Style, Width, Color, and Border Radius.

4. If necessary, select any of the following options:

- Click **Default text style** and select a text font, size, color, and style.
- Select **Required** if a Process Experience user must enter a value for the property. Select **Read Only** if a Process Experience user cannot change the value for the property.
- Select or clear **Show Label** to specify whether to show a label for the property in Process Experience. This option is selected by default. The name of the property is supplied but you can type a different name in the text box. You can also click **Default label style** to specify the text font, size, color, and style for the label.
- Select or clear **Show on mobile devices** to indicate whether to display the property on mobile devices. This option is selected by default.

5. If you want to associate the property with a category that will be disabled or hidden in Process Experience, select it and then type a name in the Category box in the right pane. See [Disabling or hiding information on a form](#).

Defining filters on form relationship controls

An application often needs to limit the items that are available for users to select when establishing relationships using a form. In many occasions, the selection needs to be dynamically filtered based on the application context.

When designing a form, you can associate a relationship control with a filter that contains context variables.

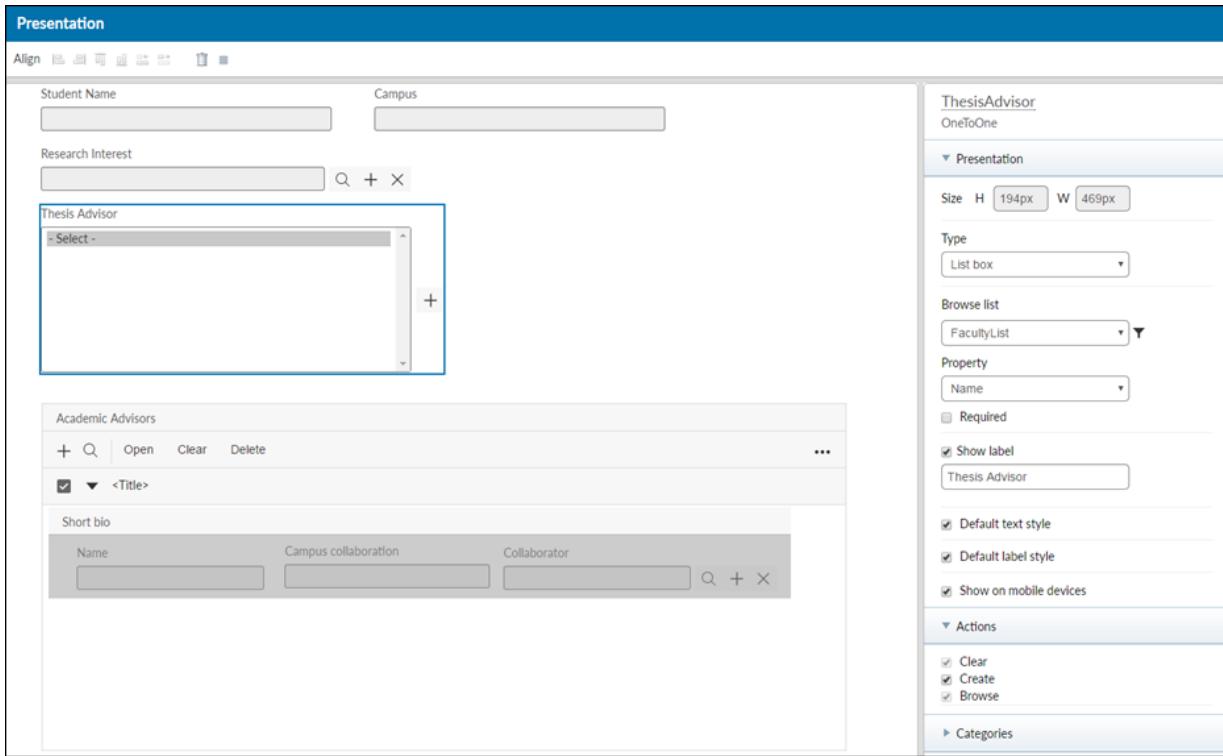
A filter can be associated with a relationship in the following cases:

- A To One relationship that uses one of the following presentations: Buttons only, Text box, Drop list, and List box.

- A To Many relationship that uses a repeating group or grid presentation.

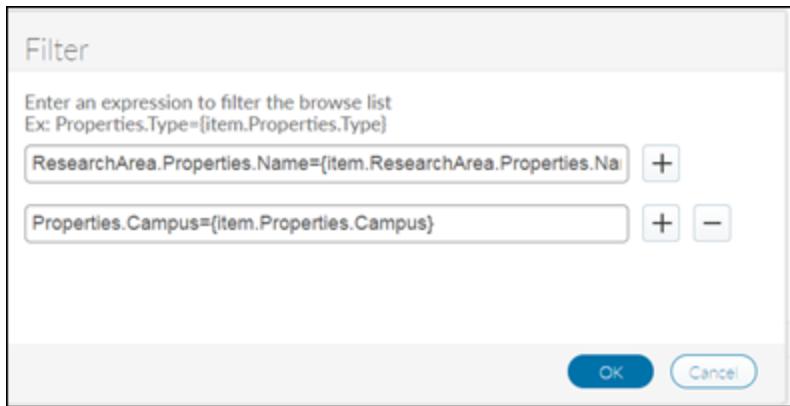
A relationship control that has an associated filter is shown with a filter icon on the Configuration panel. The filter is filled with solid black (▼) if a filter is defined or it is blank if no filter is defined.

In the following example, when selecting an advisor for a student, the advisor selection can be limited based on the research interest the user selects for the student. When you click the filter icon in the Browse list area, you can specify the filter to use.



To define a filter:

- Click the filter icon.
The Filter dialog box opens.



A filter can consist of one or more terms. Click (Plus) to add a new term or (Minus) to remove an existing term. When multiple filter terms are specified, they are logically joined together (using the AND operator) to form a complete query.

Each filter term consists of a property, operator, and value as follows:

`property=operator(value)`

The property refers to a property in the browse list. To refer to a local property in the browse list, simply prefix the property with the property group name, for instance, `Properties.Campus`. To refer to a related property in the browse list, prefix the property with relationship name, for instance, `ResearchArea.Properties.Name`.

Note: The property must be searchable in the browse list. A list property used in the filter must be defined as **Include as filter** and must not be defined as **Hide in results**.

The operator specifies how the property value should be compared with value expression. When the operator is omitted, it defaults to `eq` (equals).

The value is encoded inside parentheses `()`. A value can be either a constant value or in, most cases, an expression with variable substitution enclosed in brackets `{}`. The variable syntax is the same as what is used in other building blocks such as the [Web Content](#) panel.

A filter can include item context variables, system variables, and user variables.

In Process Experience, the item context binds to the item of the form where the relationship control is defined.

If a context variable evaluates to null, the term is ignored.

Example:

`Properties.Campus=eq({item.Properties.Campus})`

Related properties can be referenced by prefixing them with the relationship name.

Example:

`ResearchArea.Properties.Name=eq({item.ResearchArea.Properties.Name})`

Example:

`Properties.Campus=eq({item.Properties.Campus})` is the equivalent of
`Properties.Campus={item.Properties.Campus}`

The following table lists the supported comparison operators:

Operator	Description	Notes
<code>eq</code>	Property values are equal to the parameter value	This is the default operator if an operator is not specified,
<code>ne</code>	Property values are not equal to the parameter value	
<code>lt</code>	The property values are	

Operator	Description	Notes
	less than the parameter value	
gt	The property values are greater than the parameter value	
ge	The property values are greater than or equal to the parameter value	
le	The property values are less than or equal to the parameter value	
Like	The property values are "like" the parameter value, using the database server's wild card capabilities	item.Properties.StudentName=Like(A%)
InList	The property values are among a specified list of possible values	item.Properties.Campuse=InList(Berkeley,LA,SF) no spaces before or after comma.
NotInList	The property values are not in a specified list of possible values	item.Properties.Campuse=NotInList(Berkeley,LA,SF) no spaces before or after comma.
Between	The property values are between the specified bounds	item.Properties.Value=Between(13,99) no spaces before or after comma.
Null	The property values are null	Properties.Published=Null()
NotNull	The property values not null	Properties.Published=NotNull()

Working with filters defined on a list

Filters defined on form relationship controls can work together with filters defined on the list. When both filters are defined, they are logically joined (AND) together to form the complete query.

Adding containers to a form

Instead of adding components directly to a form, you can organize the information into containers. This helps to simplify the presentation, especially if the form contains a large

number of properties. For example, if you are creating an employee application you might organize the information into containers such as Education, Experience, and Skills.

To add a container to a form:

1. Create the form and define its properties.
2. In the Components panel, expand **Other** and drag **Container** to the form.
3. Select or clear **Show container heading** (the option is selected by default). If selected, the container is shown with a heading in Process Experience. By default, the heading is Container but you can type a different heading in the text box.
4. If necessary, select any of the following options:
 - Click **Default text style** and select a text font, size, color, and style.
 - Click **Default label style** and select a label font, size, color, and style.
 - Click **Default background** and select a color or image. See [Adding images and color to a form](#).
5. If you want to associate the container with a new or existing category, type its name (no spaces) in the Category box and then click **Add**. Categories can be used in rules to disable or hide information in Process Experience. See [Disabling or hiding information on a form](#).
6. Click **Default border** and select a style, width, and color for the container border.
7. Select or clear **Show on mobile devices** to indicate whether to display the property on mobile devices. This option is selected by default.
8. Drag properties to the container and configure their attributes. See [Adding properties to a form](#) and [Adding relationships to a form](#).
9. Click  (Save).

Adding tab containers to a form

By organizing the information on a form into tabs, you can present a large amount of information in an organized way. For example, if you are creating an employee application you might organize the information into containers such as Education, Experience, and Skills. Users can easily switch from one tab to another to see the information they need.

Tip: It is good practice to test the form to be sure that information fits on each tab as you intend.

To add a tab container to a form:

1. Create the form and define its properties.
2. In the Components panel, go to the **Other** list and drag **Tab Container** to the form.
3. In **Number of tabs**, type a value.
4. If necessary, select any of the following options:
 - Click **Default text style** and select a text font, size, color, and style.
 - Click **Default label style** and select a label font, size, color, and style.

- Click **Default background** and select a color or image. See [Adding images and color to a form](#).
 - Click **Default border** and select a style, width, and color for the container border.
 - Select or clear **Show on mobile devices** to indicate whether to display the property on mobile devices. This option is selected by default.
5. If you want to associate the container with a new or existing category, type its name (no spaces) in the Category box and then click **Add**. Categories can be used in rules to disable or hide information in Process Experience. See [Disabling or hiding information on a form](#).
 6. Drag properties to each tab and configure their attributes. See [Adding properties to a form](#) and [Adding relationships to a form](#).
 7. Click  (Save).

Adding hyperlinks to a form

You can add hyperlinks that enable a user to navigate directly from a form to another layout or to an external web page. You can also provide the capability to add dynamic parameters using properties of the following entities:

- Properties of the current entity {item.Properties.[PropertyName]}
- User properties {USER.Properties.[UserProperty]}
- System properties {system.[SystemProperty]}
- Properties of a related entity property {item.[Relationship].Properties.[PropertyName]}

Example URLs:

- In the following example, the two parameters are taken from the current entity.

```
https://www.google.com/maps/{item.Properties.longitude},  
{item.Properties.latitude}
```

- In the following example, the parameter is taken from the user properties.

```
https://www.google.com/?qws_rd=ssl#q={USER.Properties.PreferredLocale}
```

- In the following example, the beginning of the URL is obtained from the system properties:

```
{system.baseurl}app/processExperience/web/perform
```

- In the following example, both parameters are taken from the "Peer" relationship on the entity.

```
https://www.google.com/maps/{item.Peer.Properties.longitude}/  
{item.Peer.Properties.latitude}
```

To add a hyperlink to a form:

1. Select the form and click **Configure**.
2. Expand **Other** and select **Hyperlink**.
3. In **Text**, type the text that will identify the hyperlink to a user.
4. In **URL**, type the URL that will open when a user clicks the link text.
5. Specify where the link will open in Process Experience.
 - Current window – The current page is reloaded with the hyperlink URL.
 - New window/tab – The hyperlink URL opens in a new tab or window.
 - Modal window – The hyperlink URL pops up as a modal window on the current page. Users cannot perform any operations on the main/parent window until they close the modal window.
6. Select or clear **Show on mobile devices**.
7. If applicable, select or add a category.
8. Click  (Save).

Adding actions to a form

A form can include actions from the entity you are configuring. When a Process Experience user clicks an action button, the action is invoked. If the action requires parameters, the user is prompted to enter the parameters and passes them as part of invoking the action. The actions that are available are based on the item's state and context. You can also add actions (other than Delete and Print) from a related entity (relationship with Multiplicity To One). Related entity actions are available under the respective relationship tree. For example, you can add the **Upload** action to a Create Form to upload a file in Process Experience.

Note: For Imported entities, actions are not shown in a form when it is launched from the Create action in Process Experience.

To add an action:

1. Select the form.
2. In the Components pane, expand Action Buttons and drag the actions buttons you want to the Presentation pane.
3. Update the following properties if required:
 - A label that identifies the action. The label is editable and supports translation.
 - A tooltip that is displayed to Process Experience users. By default, this is the same as the label. The tooltip is editable and supports translation.

- A path that identifies the source of the action (building block and relationship). This information cannot be edited.
4. Select or clear **Show on mobile devices** to indicate whether to display the action button on mobile devices.
 5. If you want to associate an action button with a category, click **Categories**, type the category name in the text box, and click **Add**.
 6. Click  (Save).

Adding images and color to a form

By including images and color in your organization's forms, you can give them a distinctive look and feel that is customized to your particular requirements. For example, you can:

- Include your company or division logo directly on a form.
- Include a hyperlink to a URL on an image.
- Include a different background for the forms for each department.
- Include a background for each container on a form.
- Include a watermark as a background for a form.
- Include an image of an industry-standard form as a background.

Before you can add images to a form, you need to include them in your project or in a folder in your project. For example, you can add the images to a project folder called Images.

To make the images available in Process Experience, you must add a Web Library Definition to your project to include your image folder before you publish the project. If you later need to specify a different image folder, you must re-publish the project.

To create a folder for images (optional):

1. Right-click the project and select **New > Folder**.
2. Replace **Untitled Folder** with the name you want.

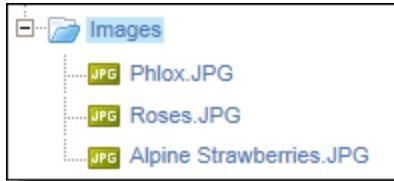
To include images in a project or folder:

1. Right-click the project or folder and select **Upload Document**.
The Upload Document wizard begins.
2. Click **Browse** and select the image file.

Note: The image browse dialog box may not recognize certain file types as images. For example, .ico and .svg files are not supported.

3. If necessary, modify the displayed description.
4. Click **Finish**.

In the following example, three images were added to the Images folder.



To add a Web Library Definition to your project:

1. Open your workspace, if it is not already open.
2. Click (New).
3. Select **Other**.
4. Type **Web Library Definition** in the search box, and then select it in the search results.
5. In Web Content Settings, browse to the location of the image folder and click **Save**.
6. Click **OK**.

To include an image directly on a form:

1. Select the form.
2. In **Components > Other**, click **Image**.
3. In the right pane, click (Browse) and select the image.
4. In **Name**, provide a name for the image.
5. If you want to include a hyperlink on the image, select **Hyperlink**.
 - a. In **URL**, type the URL for the link.
 - b. In **Open in**, select where to open the hyperlink:
 - Current window – The current page is reloaded with the hyperlink URL.
 - New window/tab – The hyperlink URL opens in a new tab or window.
 - Modal window – The hyperlink URL pops up as a modal window on the current page. Users cannot perform any operations on the main/parent window until they close the modal window.
6. Click (Save).

To include an image as a background for a form:

1. Select the form and click **Presentation**.
2. In the right pane, click **Default background**.
3. Under Image, click **Browse** and select the image.
4. Click (Save).

To include an image as a background for a container:

1. Select the form.
2. Select the container.
3. In the right pane, click **Default background**.
4. Under Image, click **Browse** and select the image.
5. Click  (Save).

To select a background color for a form:

1. Select the form and click **Presentation**.
2. In the right pane, click **Default background** and select a color from the list.

To remove a background color or image:

- Click  (Break link).

Adding text and lines to a form

You can enhance your forms with text and lines.

To add text:

1. In **Components > Other**, click **Text**.
2. Drag the text box to the location where you need it.
3. In the right pane, type the text in the Text box.
4. Click **Default text style** and select a font, size, color, and style.
5. Resize the text box on the form to accommodate the text.
6. Select or clear **Show on mobile devices** to indicate whether to display the property on mobile devices. This option is selected by default.
7. If you want to associate the container with a new or existing category, type its name (no spaces) in the Category box and then click **Add**. Categories can be used in rules to disable or hide information in Process Experience. See [Disabling or hiding information on a form](#).

To add lines:

1. In **Components > Other**, click **Horizontal Line**.
2. Drag the line to the location where you need it and resize its width based on your requirements.
3. Select or clear **Show on mobile devices** to indicate whether to display the property on mobile devices. This option is selected by default.
4. If you want to associate the container with a new or existing category, type its name (no spaces) in the Category box and then click **Add**. Categories can be used in rules to disable or hide information in Process Experience. See [Disabling or hiding information on a form](#).

Designing responsive forms

When designing some forms, you place controls of a specific size in specific locations on the form canvas. In your application, these controls are shown in the same size and location as designer. This rendering method is required for certain types of applications, for instance when form is overlaid on an image.

However for other applications with large and complex dynamic forms, you will typically want those forms to use the space more efficiently. For example, you may have the following requirements:

- Close up vertical white spaces created by hidden controls.
- Enable repeating group containers to automatically grow or shrink in height as needed, based on content, and automatically move the controls below repeating group containers up or down.
- Configure responsiveness based on the positioning of the controls.

Sections

A section on form is a horizontal region that extends the entire width of the form canvas.

You add a new section to divide form canvas into horizontal regions. You can then add components to a section in the same way that you add them to a container on a form. However, unlike a container, a section provides two important capabilities for creating responsive forms:

- When every component in a section is hidden, either by rules or security, the section is hidden and the white space left by the hidden controls goes away.
- By responding to the visibility and height changes of adjoining sections, all controls in a section move up or down so that the form becomes shorter or longer responsively.

A section also has the following characteristics:

- Sections are top level components in that they cannot be nested inside containers.
- By default a new form is not responsive until you add a section.
- Sections are stacked from top to bottom in the form designer and in your application. Sections do not have an absolute position to themselves but an index that states which section is on top of which, and they are stacked in that order in the form.
- Sections do not have a scroll bar.
- Components are absolutely positioned inside a section.
- In your application, a section is hidden when all controls inside it are hidden and the space it occupied is not shown.

Autosize repeating group container

The content of a repeating group depends on the number of rows in the container. The height of a repeating group container grows or shrinks based on the rows available.

You must explicitly configure a control to automatically grow or shrink inside the section where it resides. When a control is set to autosize, it generally makes the best sense to place it as the lowest positioned control in the section.

To ensure the proper movement when the height of the autosize control changes, place controls that need to be positioned below it inside a separate section.

You can set a maximum size to limit the amount of screen space an auto-sizable control occupies. If no maximum is set, controls can grow indefinitely.

In the Form Designer window, a toolbar is available for creating a section. Each section is separated by a dotted line.

To create a section:

- On the toolbar, click Icon Image (Insert section before) or icon image (Insert section after).

To move a section:

- Select the section and then click Icon Image (Move section up) or Icon Image (Move section down).

To remove a section:

- Use the same procedure that you use to remove containers and components on form. When a section is removed, all components inside it are also removed.

To auto-size a repeating group container:

1. Place it in a section.
2. Select **Auto-resize as items are added or removed**.
3. Optionally, specify the number of items to display before providing a vertical scroll bar.
4. Configure the remaining options as described in [Adding a To Many relationship to a form](#).

Sample form

The following screenshot shows an example of a responsive form in which the repeating group container has no whitespace and no scroll bar.

The screenshot shows a form interface with the following fields and sections:

- Top Row:** Buttons for Print and Delete, followed by fields for Id (32770), Urgency (NotSpecified), and Assignee (- Select -). To the right are radio buttons for Show Related (Yes) and Show Watchers (Yes).
- Description:** A large text input field.
- Show Related:** A section with a radio button for Yes (selected) and a radio button for No.
- Related Tickets:** A section titled "RelatedTickets" containing a search icon (+ Q) and a list with 1 item. The list includes an expandable row for ticket 32769 with fields for Id (32769) and Description.
- Show Watchers:** A section with a radio button for Yes (selected) and a radio button for No.
- Watchers:** A section titled "Watchers" containing a search icon (+ Q) and a list with 1 item. The list includes an expandable row with fields for Name and Email.
- Comments:** A large text input field.

Disabling or hiding information on a form

By disabling or hiding certain parts of a form, you have great flexibility for customizing it for different types of users. For example, assume that you create a form that contains properties called Apples and Bananas. If the value of Apples is A, you want to disable the Bananas field.

Tip: If you want to prevent a user from replacing a related property with another value, delete the Search icon from the form. If you want to prevent a user from changing a property based on a particular condition, assign the property to a category and then create a rule defining the condition under which the property or category will be displayed or hidden.

To disable or hide information on a form:

1. Create the properties that you plan to include on the form.
2. Create a form that includes the required properties.

3. Create a rule that defines what to do when the conditions of the rule are true. See [Adding rules](#).

In the following rule, when the value of Item = Laptop, the property called Color is disabled. To disable a category, you would select Category instead of Property.

Rule Properties

When:
A property changes

If: all of the following are true
Item equal to Laptop

Then:
Disable
Property Color

4. Create a Full layout that includes a Form panel and select the form in the panel's properties.
5. Click (Save).
6. Publish the project.
7. Test the rule.
 - a. Run Process Experience.
 - b. Click **Create**.
 - c. Select the form that you created.
 - d. In the Item field, type Laptop. The Color field becomes disabled.

Nesting forms

Each form that was created for an entity or its related entities is available for nesting within the form you are currently creating. Using nested forms can save a considerable amount of time if an entity has many forms that include the same groups of components. For example, you can create a "mini-form" that contains properties such as Name, Title, Street, City, Postal Code, and Email Address and components such as text boxes and tab containers. That form is then available for you to drag to other forms that you create or edit for that entity. By creating a library of mini-forms that you intend to nest in other forms, you can save a considerable amount of time when creating forms.

Note: You cannot nest forms for child entities in Create forms.

Instead of dragging nested forms directly to a form, another option is to create tab container and then drag each form to a different tab. For example, assume that you create

the following forms for Job Candidate entity: Personal Information, Work Experience, and Certifications.

In your Default form, you include a tab container with three tabs and then you nest each form in a separate tab. When a hiring manager selects a Job Candidate from a list, all of this information is available in a single location.

To nest a form, you simply drag it to the Presentation panel. When you do this, the nested form is shaded to indicate that it is linked to the source form and the right panel displays the name of the form to which it is linked. Any changes made to the source form are reflected in the linked form. You cannot edit a nested form that is linked.

You can break the link by clicking **Break link** in the properties panel on the right. When the link is broken, you are working with a copy of the source form. Any changes made to the source form are not reflected in the nested form and you can edit it to customize it to your requirements. Any changes you make to the nested form will not be included in the source form or any other forms that are linked to it.

Note: In a To Many relationship container (repeating group container), if a subform (of the related entity) is added and you break the link to it, the categories and rules defined on it are no longer triggered. This is because the categories and rules were configured on the subform and the related entity rather than the main form and the source entity.

Example

An entity representing a user account might contain the following properties:

- User Name
- Password
- First Name
- Last name
- Street Address
- Apartment or Unit number
- State
- Zip Code
- Country
- Email

Assume that the application requires the following forms:

- Create - to create an account
- Private Profile - to edit all account properties
- Public Profile - to display a subset of account properties

One approach would be to place all of the required properties directly on each form. A better approach might be to define the following smaller forms, which could be then re-used as nested forms in the three "main" forms, and potentially in other forms:

Credentials might contain the following properties:

- User Name
- Password

Full Name might contain the following properties:

- First Name
- Last Name

Full Address might contain the following properties:

- Street Address
- Apartment or Unit number
- State
- Zip Code
- Country

The Private Profile form can then include Credentials, Full Name, and Full Address as nested forms.

The Create form can just include the Private Profile form, with the header and border turned off for better appearance.

The Public Profile form can include Full Name and Full Address as nested forms.

Assume that the Create form in Process Experience cannot fit the Private Profile form vertically in its entirety. To correct this, you can use the Break link option on the Private Profile form inside the Create form to separate this particular Private Profile form instance from its definition and rearrange the nested forms horizontally. After breaking the link, any subsequent edits do not affect the initial definition, only the selected nested form.

Clicking **Break Link** turns the Private Profile form into an editable adjustable container. Its inner components can now be repositioned or deleted and new components can be added. The inner nested forms remain read-only since they represent separate forms. Links are not broken recursively and automatically. If needed, the inner links can be further broken for each individual nested form.

A Create form in which the Private Profile form was adjusted might appear as follows in Process Experience.

The image shows a user interface for creating a new entity. It consists of three vertical columns. The first column, titled 'Credentials', contains two input fields: 'User Name' and 'Password'. The second column, titled 'Full Name', contains two input fields: 'First Name' and 'Last Name'. The third column, titled 'Address', contains four input fields: 'Street Address', 'Apt/Unit', 'Zip', and 'State' with a dropdown arrow. At the bottom left are two buttons: 'Create' and 'Cancel'.

Positioning components on a form

You have the following options for positioning components on a form.

- To move a component, drag it to another location on the Presentation panel or use the [Keyboard shortcuts for forms](#).
- To resize a component, drag its sizing handle or borders or use the [Keyboard shortcuts for forms](#).

Note: The height of the following components is automatically resized based on the font and you can resize only their width: Textbox, TextOnly, MultiLine, Date, DateTime, Checkbox, Dropdown list, Listbox, and Radio Buttons. You also cannot resize the height of the Horizontal Line.

- To align components, click (Align Left), (Align Right), (Align Top), and (Align Bottom) on the toolbar.
The selected components are aligned based on the first component that was selected.
- To resize multiple components, click (Fit to Shortest Width) and (Fit to Longest Width) on the toolbar.
The components are resized based on the width of the shortest or longest component.
- To delete the selected components, click (Delete selected components) on the toolbar.
- To toggle the display of the alignment grid, click (Toggle Grid) on the toolbar.

Note: To enable creating forms with multiple columns, the alignment operations intentionally align components relative to the first component selected rather than to the workspace.

Keyboard shortcuts for forms

When designing forms, the following keyboard shortcuts are available for commonly used functions.

Shortcut Key	Function
←, →, ↑, ↓	Move the selected components to the next five pixel boundary in the specified direction.
CTRL+←, CTRL+→, CTRL+↑, CTRL+↓	Move the selected components one pixel in the specified direction.
SHIFT+←, SHIFT+→	Increase or decrease the width of the selected components by 5 pixels by moving the right edge of the component.
SHIFT+↑, SHIFT+↓	Increase or decrease the height of the selected components by 5 pixels by moving the bottom edge of the component.
ALT+←, ALT+→, ALT+↑, ALT+↓	Align multiple selected components left, right, top and bottom. Use the left and right arrows to align components that are vertically positioned on the palette. Use the up and down arrows to align components that are horizontally positioned on the palette.
CLICK and drag (on a component or container)	Moves the component or selected container, aligning its top right corner to a 5 pixel boundary.
CTRL (while dragging)	Enables moving components to any position (normal dragging moves components to five pixel boundaries).
CLICK and drag (on a container that is not selected)	Shows a selection rectangle that, when the mouse button is released, selects all the components that are inside the rectangle.
CLICK	Selects the selected component and deselects all other components.
CTRL+CLICK	Toggles the selection of the component. This does not change the selection state of any other components. If the component is selected, it becomes the current master.
SHIFT+CLICK	Selects the component without changing the selection state of any other components.
ESCAPE	Cancels any dragging operation in progress.

Tracking form progress

You can configure progress bars to track the progress on a form, such as how much of it has been completed. The progress bar calculates the progress based on the number of elements the form contains and the weight of each element. Progress bars are available for numeric property types such as Integer, BigInteger, Short, Decimal, Float, and Double. Progress bars are also supported in tab containers, repeating group containers, and grids.

To create a progress bar property:

1. Open the entity.
2. Create a property. In this example, the Display Name is Form Completion and the name is FormProgress
 - a. In **Property Type**, select a numeric property type (such as Integer, BigInteger, Short, Decimal, Float, or Double).
 - b. In **Max**, specify a maximum value based on the number of elements in the form and the weight of each element. For example, if the form has 5 elements, and each element has a weight of 1, the max should be 5.
 - c. Save the property.

To create a rule that will govern the progress value as a user fills out the form:

1. Create a rule and click **Configure**.
2. In **When**, select **When a property changes**.
3. Specify the following conditions (select the display name of the property you created for the progress bar).



4. In **Then**, click **Set**.
5. In **Target**, type Item.Properties.<name of the propgess bar property>
6. In **Source**, define the form elements to track using the following format:

```
(item.Properties.fName is not null ? 1 : 0) +
(item.Properties.address is not null ? 1 : 0) +
(item.Properties.DOB is not null ? 1 : 0) +
(item.Properties.lName is not null ? 1 : 0) +
(item.Properties.SSN is not null ? 1 : 0)
```

- The `Item.Properties.<property name>` denotes the name of the element. For example, `Item.Properties.DOB` denotes the DOB property on the form.
- The `1:0` designates the weight of the element. If the element is completed (is not null), the weight is shown in the number preceding the colon. If the element (such as DOB) has not been completed (is null), the weight is shown in the number following the colon. The sum of the weights for all elements must match the Max value for the progress bar property. The default value is 100.

Result: If the form has 5 tracked properties, each property has a weight of 1, and only the first 2 are filled out, this rule produces a value $1 + 1 + 0 + 0 + 0 = 2$. If the progress bar has a Max setting of 5, the resulting progress displayed will be $2 / 5 = 40\%$

7. Save the rule.

To configure the form:

1. Open the form and click **Configure**.
2. Drag the progress bar property to the form.
3. Select the progress bar property.
4. Expand **Presentation**.
5. In **Type**, select **Progress bar**.
6. Save the form.

Tip: The preceding sections use a simple example to explain how to track the progress of a form. By configuring more sophisticated rules you can provide more advanced progress bar tracking. For example, track progress towards a charitable contribution target. Each property would contain a dollar amount to be committed to that target. Get the sum of each property value, and the progressBar property Max would be the target.

Drop list series example

In this example, a drop list series is created using multiple related entities. A Process Experience user creates a project, selects a product for the project from a drop-down list, and then selects a product line from a second drop-down list whose values are filtered based on the product that was selected.

To achieve this, the project needs three entities:

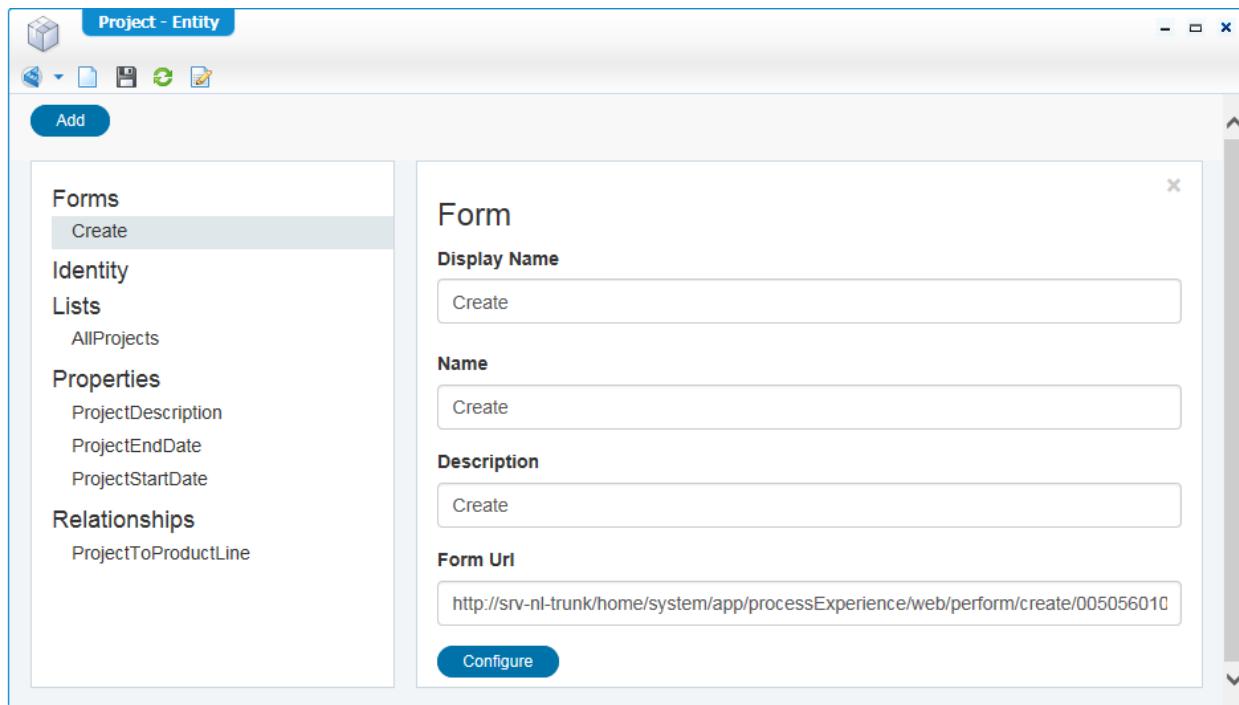
- Project
- Product
- ProductLine

Note: The examples given in this section are for demonstration only and specific to a particular use case. You may need to adapt the procedures so that they are relevant for your application and domain model. When using a drop list series in a production application, you will typically need to include many additional building blocks.

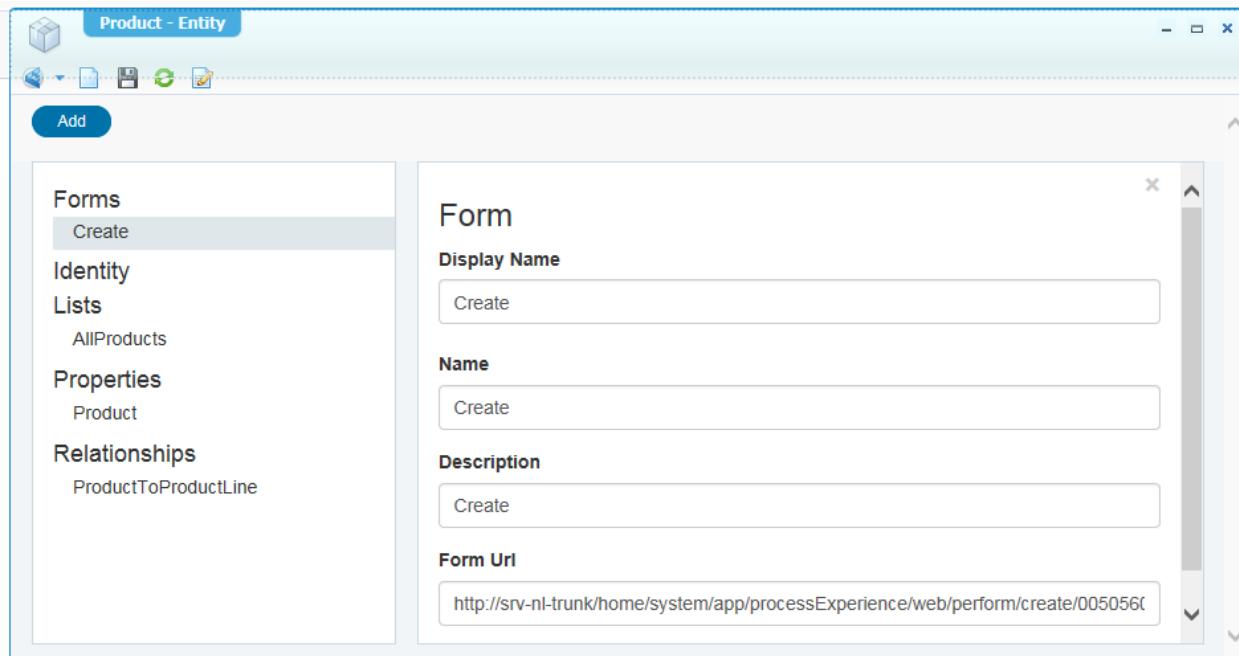
Entity configuration

Each of the entities must have properties, forms, lists, and relationships, as shown in the following screenshots.

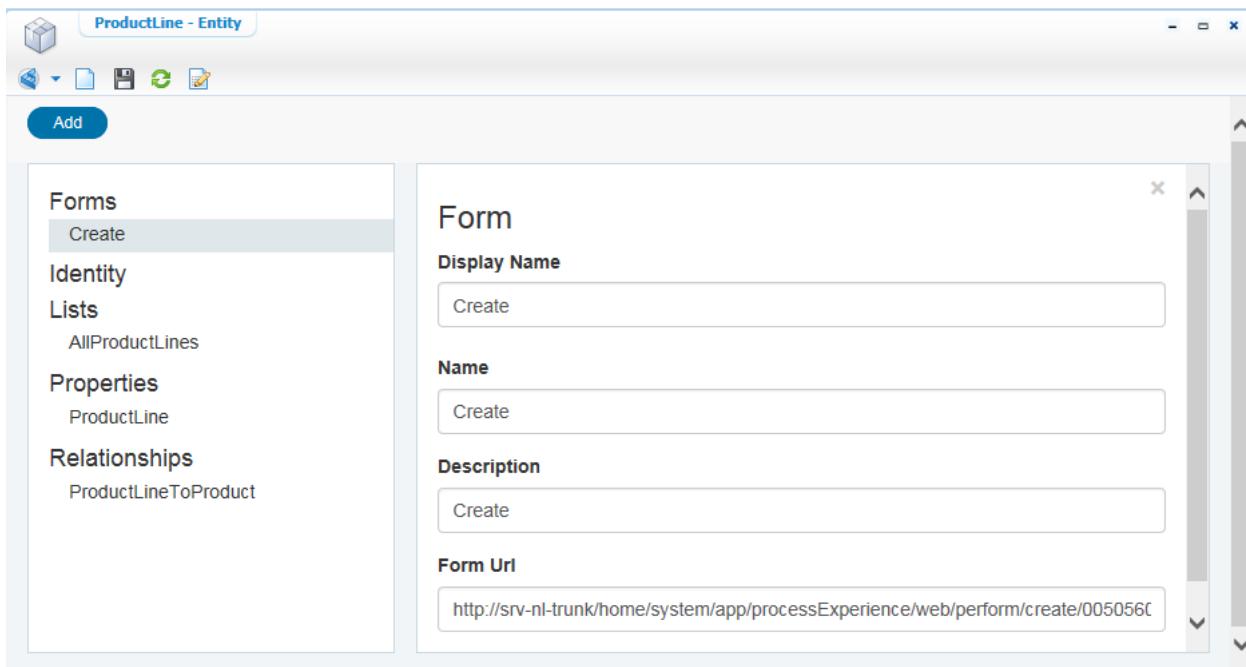
Project entity



Product entity



ProductLine entity



Building block configuration

The following steps are required to get the desired results with a drop list series.

Important: You must perform the steps in the order in which they are presented.

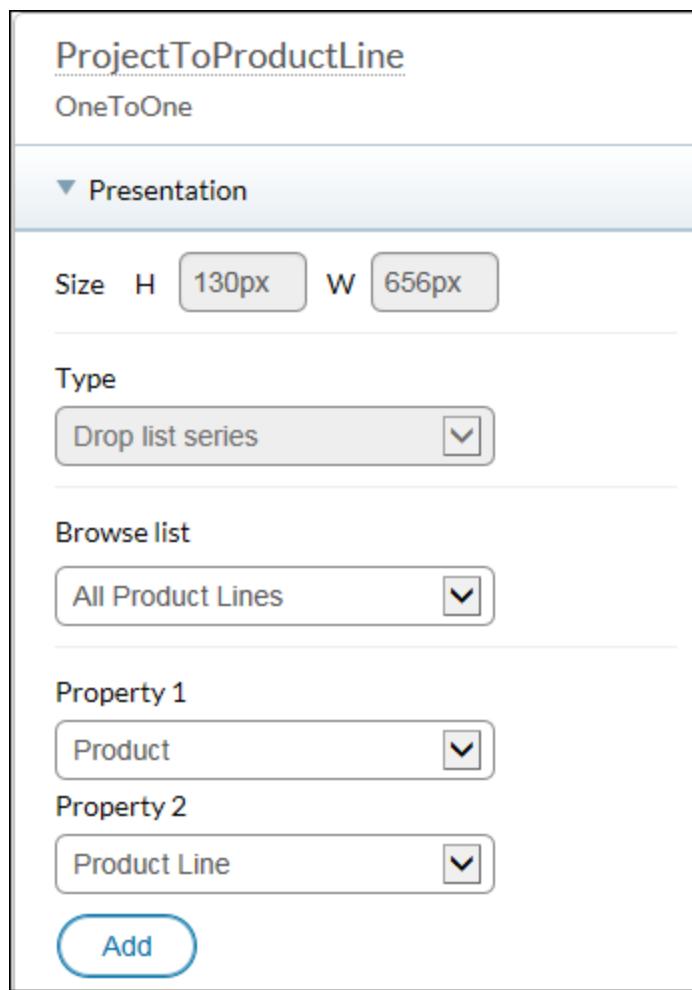
To configure the building blocks for a drop list series:

1. Add the properties:
 - The Project entity has properties called ProjectDescription, ProjectStartDate, and ProjectEndDate.
 - The Product entity has a property called Product.
 - The ProductLine entity has a property called ProductLine.
2. Build the relationships. Keep in mind that the user will create a project and then select the Product and Product Line.
 - The Product entity has a To Many relationship with Product Line. (Each product can have many product lines.)
 - The ProductLine entity has a To One relationship to Product that is paired. (A product line can have or be associated with only one product.)
 - The Project entity has a To One relationship to Product Line.
3. Configure the lists.

All entities require a list. In the ProductLine entity, be sure that both Product and Product Line properties are displayed in the list. This is important because these are the two

properties that are needed in the drop list series. The properties that will be available for selection must be included in the list.

4. Design the forms. In addition to building a Create form, you can also build a Default form for each entity to view items.
 - a. In the Project entity, edit the default Create form or create a new Create form as follows.
 - Drag the To One relationship called ProjectToProductLine to the form and change the presentation to Drop List Series.
 - Select the Browse List (in this case, the list is called All Product Lines).
 - Select the Product property and click **Add** to add the Product Line property.



- b. In the Product and ProductLine entities, add Create forms. In Process Experience, users will not see any products or product lines in the drop down lists if they haven't created any products or product lines. You can build the Product Line Create form with something similar to the following example.



5. Publish the project and, if necessary, assign access rights in the Administrator tool.

Process Experience flow

In Process Experience, the flow will be something like this.

1. Create one or more products.
2. Create one or more product lines for the product.
3. Create a project and select the Product and Product Lines from the drop down lists.

Create Project						
Project Description	Project Start Date	Project End Date				
Inventory	02/20/2017	02/22/2017				
ProjectToProductLine <table border="1"> <tr> <td>Product</td> <td>Product Line</td> </tr> <tr> <td>Computer</td> <td> - Select - Desktop Laptop Notebook Tablet </td> </tr> </table>			Product	Product Line	Computer	- Select - Desktop Laptop Notebook Tablet
Product	Product Line					
Computer	- Select - Desktop Laptop Notebook Tablet					

Adding lists

Use the List building block to configure lists that will enable users to display and find items in Process Experience. Lists are the most common way for a user to interact with the system. A user accesses a list to display items and then opens or works with the items.

Note: You cannot include properties of an imported entity (an entity imported from an existing database schema) in the list of a related native entity (an entity modeled in Process Suite). See [Importing entities from database tables](#) and [Importing entities from other applications](#). This also applies to EIS entities (entities that are generated from an EIS Connector). See [EIS connectors](#),

To add a list to an entity:

1. In **Workspace Documents**, open the entity.
2. Click **Add > List**.

3. In the List Details pane, provide a Display Name and Name.
 4. In **Default Category**, specify an (optional) default category for the list in Process Experience.
Lists are created in the default category for a user who is accessing the system for the first time but users can rearrange the lists into other categories that satisfy their requirements.
 5. In **Display Count**, select an option for how to display the number of available items that will be returned by the list: **Never**, **Only in results**, **In lists and results**.
 6. In **Results per page**, select **All** or specify the maximum number of items to show in the list.
If the list returns more than this number of results, the user can page through them.
 7. If you want the list to be visible to Process Experience users, select **Visible to end user**.
 8. In **Action Bar**, select an action bar to be shown for the list.
In Process Experience, any actions that are not relevant are not available. See [Creating an action bar](#).
 9. In **Full Layout**, select a full layout for the list (optional).
When a user opens this list record in your application, the selected layout will be displayed. If you do not select a full layout, the list opens based on the configured behavior.
If the entity contains a lifecycle building block and a list is defined on the lifecycle task entity:
 - Even if you select a full layout, the layout configured in the lifecycle activity is displayed.
 - If a layout is not configured for the lifecycle activity, the selected full layout is shown when the task is opened.
 - If a layout is not configured for either the lifecycle activity or the list, layout is displayed based on the existing behavior.
 - For lifecycle BPM activities, selecting a full layout for the list does not affect the existing behavior.
 11. Click **Add > Configure**. See [Configuring a list](#).
- Tip:** If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.
12. Click  (Save).

Configuring a list

After you add a list, you need to select the properties to include and the filters to apply when the list is accessed. These rules limit the items that users are able to access. A user does

not see any items that do not match the list's filter rules. A user cannot change or disable these filter rules.

If the list includes any filterable properties, it displays a filter form where a user can specify additional filters for the list. In general, leaving a filter field blank means that the list will not depend on the property's value. The user filter form is quite powerful because the user can choose the comparison operators to be used.

Note: A list's filter rules may not be available to a Process Experience user if the list references a property that the user is not authorized to know about.

The value of a property in the filter term can be a constant or an expression that you type in the text box that is displayed when certain operators are selected.

An expression must be formatted as follows:

`$(yourExpression)`

For example:

List property	Operator	Expression Value	Description
Grade	any of	<code>\$({"A", "B", "C"})</code>	Finds all students with a grade of A, B, or C.
assignee	equal to	<code>\$(User.Properties.UserId)</code>	Finds all instances where the value of assignee is the same as the UserId of the current user.
contracttype	any of	<code>\$(getContractTypes())</code>	The Contract Center solution uses the expression support to get a list of contract types for the current user via script function <code>getContractTypes()</code> .
Due Date	equal to	<code>\$(today())</code>	Find items where the due date is today.

Notes:

- Date, DateTime, Boolean, Enumerated Text, and Enumerated Integer properties do not allow specifying the expression as a value.
- The format of the **any of** operator where the value is a constant is a sequence of strings separated with semicolons as in red; white; blue. If a semicolon needs to be part of one of the strings, that string is enclosed in single quotation marks. The `$(variable)` form is used only for expressions.

To configure a list:

1. Add or open the list and click **Configure**.
2. In **Available Properties**, select the properties to include in the list.
In **Properties shown in results**, the properties are listed in the order in which you

select them but you can use the following options to rearrange them: Expand All, Collapse All, Remove, Move Up, Move Down.

Note: The Duration property cannot be used for searching and cannot be included in a list.

3. On the Properties tab, specify the following information for each selected property:
 - Column label - By default the label of the result column is the property's display name. However, labels designed to work well in a form are often too long for a column header. Using this option, you can enter an alternate (usually shorter) label for the column.
 - Column presentation - Represents the presentation type of the column in the list. The available options depend on the property type. The Column presentation option is available for Integer, Decimal, Float, Enumerated Text, Enumerated Integer, and Currency property types.
 - For Boolean, static Enumerated Text, and static Enumerated Integer property types the following presentation options are available: **Display Name**, **Icon**, and **Icon and display name**.
 - For Integer, Decimal, and Float property types, the following presentation options are available: **Progress bar** and **Value**.
If you select **Progress bar**, the value is shown as a progress indicator with a percentage value. The percentage of the column value is calculated based on the min and max values in the property configuration. If min and max values are not specified, by default 0 and 100 are considered as the min and max values. For example, assume that you configure a property with a Min value of 0 and a Max value of 50. If a Process Experience user specifies a value of 20 for an item, the progress bar length is adjusted to 40%. See [Tracking form progress](#).
If you select **Value**, select **Left**, **Right**, or **Center** to specify how to align the values in the column.
 - For Currency property types, a **Default currency format** option is available. When this option is selected (the default), the currency presentation format in your application is derived from user's locale. You can customize the currency format by clearing the option and then setting the following options for the currency presentation:
 - Positive values - Placement of the currency symbol for positive values.
 - Negative values - Placement of the currency symbol for negative values and specifies whether the negative sign is represented by parentheses or the negative sign.
 - Digits grouping - Number of integer digits that appear in a group.
 - Digit separator symbol - String that separates groups of integers.
 - Decimal separator symbol - String that separates integer and decimal digits.
 - Column alignment - Select **Left**, **Right**, or **Center** to specify how to align items within the column.

- Sorting - The default sort order for the property: **None, Ascending, Descending**. A user can click a column heading to re-sort the results. The sort option interacts with the order of the properties in the list. The first property with sort enabled is the primary sort, the second is the secondary sort, and so forth.
 - Include as filter - Whether the property should be included in the filter form.
 - Hide in results - Whether the column should be hidden in the Results panel. This is useful when a property is included for filtering, but is not useful to include in the display.
4. On the Filters tab, configure filter rules that will be applied to the list.
- Specify whether to show results if One or All of the terms are true.
 - Define the terms by selecting a property, operator, and value.
 - To define additional terms, click  (Plus).
 - To nest terms, click  (Nest).
 - To remove terms, click  (Minus).
- Note:** See [Using advanced functions](#) for a list of advanced functions that you can use to specify a filter for a list.
5. Click  (Save).

Filter operators

On the Filters tab, the set of available comparison operators depends on the property's type, as shown in the following table.

Operator	Boolean	Date	Date Time	Decimal	Float	Integer	Long Text	Text	Enum Text	Enum Integer
Equal	X	X		X	X	X	X	X	X	X
Not Equal	X	X	X	X	X	X	X	X	X	X
Greater Than		X	X	X	X	X				
Greater Than or Equal		X	X	X	X	X				
Less Than		X	X	X	X	X				
Less Than or Equal		X	X	X	X	X				
Between		X	X	X	X	X	X	X		
Within		X	X							
Contains							X	X		

Operator	Boolean	Date	Date Time	Decimal	Float	Integer	Long Text	Text	Enum Text	Enum Integer
Any Of				X	X	X	X	X		
Empty	X	X	X	X	X	X	X	X	X	X
Not Empty	X	X	X	X	X	X	X	X	X	X

Defining lists on a tasks child entity

You can define lists on a tasks child entity. For various historical reasons, assignment values used in these tables are not consistent with the way users, roles, teams, and lists are identified elsewhere in the system.

The system provides several script functions to enable filtering in this entity.

Note: When defining a list on a task child entity use the property "TaskTarget" with the following Script functions.

Script function	Description
targetUser()	The identifier for the current user
targetRoles()	The list of roles the current user belongs to
targetTeams()	The list of teams the current user belongs to
targetWorklists()	The list of worklists the current user belongs to

These functions can be used with the syntax \$(-script-expression-) as described above. The following sections show the filters used in various inbox lists.

Personal tasks

All

Target Type = User

One

Delegated To = \$(targetUser())

Task Owner = \$(targetUser())

One

State = Created

State = Assigned

State = InProgress

State = Suspended

State = Paused

Using Boolean notation this is: Target Type = User && (Delegated To = \$(targetUser()) || Task Owner = \$(targetUser())) && (State = Created || State = Assigned || State = InProgress || State = Suspended || State = Paused)

Role tasks

All

Target any of \$(targetRoles())

State != Completed

Team tasks

All

Target any of \$(targetTeams())

State != Completed

Worklist tasks

All

Target any of \$(targetWorklists())

State != Completed

Conclusions

Note the use of the “any of” operator for these three filters, this is because the result of the function is a list.

The States are inclusive for the personal tasks list and exclusive for the other lists. To combine these into a single list, you could use the following simplified filter:

All

State != Completed

One

All

Target Type = User

One

Delegated To = \$(targetUser())

Task Owner = \$(targetUser())

Target any of \$(targetRoles())

Target any of \$(targetTeams())

Target any of \$(targetWorklists())

Note: When defining a list on a task child entity use the property “TaskTarget” with the following Script functions.

Script function	Description
targetUser()	The identifier for the current user
targetRoles()	The list of roles the current user belongs to
targetTeams()	The list of teams the current user belongs to
targetWorklists()	The list of worklists the current user belongs to

Performance considerations

To improve performance for lists, a database administrator can create indexes on columns. Indexes provide faster access to data for operations that return a small portion of a table's rows.

In general, an index should be created on a column in any of the following situations:

- The column is used in filter criteria frequently.
- A referential integrity constraint exists on the column.
- A UNIQUE key integrity constraint exists on the column.

Creating an index on columns that do not match one of these situations can decrease performance. This is also relevant for Find web service operations.

Adding a title

Use the Title building block to add a standard title to an entity. A title provides a name that enables users to easily identify items. The title building block is used by other building blocks when they need a user accessible way to identify an item. For example, the History building block uses the title. You can also add the title to a form. This building block does not add any actions or permissions to the entity.

A title can be user-defined or it can be a read-only title that is supplied by the system.

To add a title:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Title** from the list of building blocks.

Note: The Name is supplied and cannot be changed.

3. In **Create By**, select **system** or **user** to specify whether the title will be system-generated or entered manually by users.

Note: Changing the Create By option changes the behavior and properties of the entity. You should not change the option if you have deployed and created items. If you have already included the Title property in a list or form and change the option you will have to add it again.

4. If the title is system generated, in **Title Specification**, type the template to use to construct the title.

The template can use properties of the entity such as: {item.Properties.<property name>}

For example, the title specification of an entity representing an insurance claim could have the following template:

Claim by {item.Properties.LastName}, {item.Properties.FirstName} on {item.Properties.AccidentDate}

Upon creation of an insurance claim, the properties used in the template are replaced with real values. The title may then look like: Claim by Johnson, Lucy on April 24, 2016.

6. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

7. Click  (Save).

Creating an action bar

Use the Action bar building block to create one or more action bars that display various option buttons. Actions become available for adding to an Action bar in the following ways:

- The Identity building block that is automatically created for every entity makes the Open, Delete, and Print actions available.
- Adding a File building block to an entity makes the Upload and Download actions available. See [Adding files](#).
- Adding a Discussion building block makes the Add Comment action available. See [Adding a message board](#).
- Adding a History building block makes the History action available. See [Tracking history](#).
- Creating a rule that sets a property or starts a process makes the Display Name for the rule available as an action. For example, you can create a rule that sets the status of a vendor to Inactive. If the Display Name of the rule is Inactivate, the caption for the action button is Inactivate. See [Configuring a rule that defines an action button](#).
- Adding a Lifecycle building block automatically creates a child entity called Task. The Task child entity contains available actions that you can include on an action bar that is used to work with tasks.
- You can also add actions (other than Delete and Print) from a related entity (relationship with Multiplicity To One).

There are two ways to make an action bar available in Process Experience.

- When you include an Actions panel in a layout, you select the action bar to display in that panel in Process Experience. For instance, the actions panel can display actions such as Open and Delete.
- When you create a list, you select the action bar to display in the Results panel for the list.

Note: By default, the label for each action button is used as a tooltip in Process Experience. See [Renaming action buttons and drop list labels](#).

To create an action bar:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Action bar**.

3. In the **Action Bar** panel, type a **Display Name**, **Name**, and **Description**.

4. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

5. In the **Action Bar** panel, click **Configure**.

6. In **Available actions**, select the options to include in the action bar.

Each selected option moves to the Added actions list.

- To reorder buttons use drag and drop or click the left and right arrows.
- To remove a button, select it and click **Remove**.

7. Click  (Save).

Grouping action buttons

You can organize the buttons on an action bar into groups. This can be very helpful to users if an action bar provides a large number of actions of different types. For example, you might place actions such as Save, Open, and Delete in one group and actions such as Upload and Download into another group. You can group actions either into a button bar or into a drop list. If you group them into a drop list, you need to provide a label for the group. The label must be unique within the action bar.

To begin:

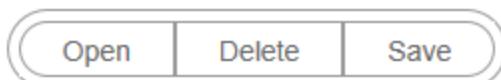
- Create a new action bar or open an existing action bar and click **Configure**.

To group action buttons into a button bar:

1. In the Added actions window, select each action button to include in the group.

2. In **Group/Ungroup**, select **Group selected into button bar**.

An example of a button bar follows:



To group action buttons into a drop list:

1. In the Added actions window, select each action button to include in the group.

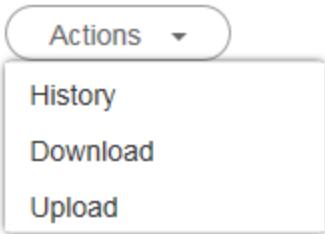
2. In **Group/Ungroup**, select **Group selected into drop list**.

The Drop list properties window opens.

3. Type a label for the drop list and click **OK**.

The label for each drop list in the action bar must be unique.

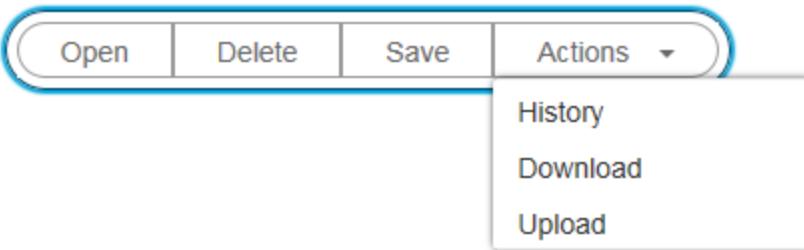
An example of a drop list follows:

**To ungroup action buttons:**

1. In the Added actions window, select the group that contains the action buttons you want to ungroup.
2. In **Group/Ungroup**, select **Ungroup**.
The action buttons are ungrouped.

To group a drop list into a button bar:

1. In the Added actions window, select the button bar group and the drop list group.
2. In **Group/Ungroup**, select **Group selected into button bar**.
An example of a drop list grouped with a button bar follows.



You can also perform the following functions:

- Move action buttons between groups by dragging them.
- Move action buttons within a button bar or drop down list using the arrow keys.

Renaming action buttons and drop list labels

Each action button has the following properties:

- A label that identifies the action. The label is editable and supports translation.
- A tooltip that is displayed to Process Experience users. By default, this is the same as the label. The tooltip is editable and supports translation.
- A path that identifies the source of the action (such as building block and relationship). This information cannot be edited.

To rename an action button:

- Select the button and click **Properties**.
- Change the label or tooltip and click **OK**.
The label and tooltip do not need to match.

To rename a drop list label:

- Select the drop list and click **Properties**.
The Drop list label window opens.
- In Label, click  (Delete) to clear the existing value, type the new label, and then click **OK**.

Adding a mobile app

Use the Mobile App building block to display a list and layout for an entity through a mobile application in AppWorks. AppWorks is the OpenText developer platform that is designed to accelerate the path from need to solution for IT organizations. With AppWorks, you can create, deploy, and manage applications that connect to OpenText services from all platforms.

You can add the Mobile App building block to an entity to generate the mobile application in AppWorks. For example, a service engineer may have a list of customer calls with defined work to be used on a mobile device while on the road. The mobile application will include a work list for the service engineer to perform the required actions, such as job completion, inspection, and modification. The required actions are available in a simple layout panel of the mobile device.

You can create multiple Mobile App building blocks for an entity and add new Mobile App building blocks to that entity at any time. For example, a Claim Management System might use multiple mobile applications: one to log and track a claim, and another for approvals from a manager.

The following functionality that you design in the entity is available in the mobile app:

- Present a list.
- Provide actions to the entity from the list (action bar and action sheet methods).
- Provide capability to create a new entity instance.
- Enable the display of actions within the item layout.
- Display actions from the Lifecycle building block that transition between states.

Note: Some actions are restricted in the mobile app. Content management actions, Lifecycle task actions, and opening documents in Brava are not supported.

The management of apps within the Gateway is covered in the AppWorks documentation.

Configuring Appworks Gateway per instance

It is possible to add the Mobile App building block to any entity. However, to publish the project, you must first configure the Appworks Gateway that the mobile applications are deployed to.

To configure the Appworks Gateway:

1. Click **Start > OpenText Process Suite Platform 16 > <your instance name> > Tools > Management Console**.

2. Click **Platform Properties**.
3. Configure the following properties:
 - **com.opentext.appworks.gatewayUrl** - The AppWorks gateway URL.
 - **com.opentext.appworks.username** - The username to deploy to the AppWorks gateway.
 - **com.opentext.appworks.password** - The password to deploy to the AppWorks gateway.
The password must be base64 encoded. There are resources and tools online to do this (Notepad++ for example).
 - **com.opentext.appworks.OTDSAddress** – The location of OTDS.
 - **com.opentext.appworks.OTDSResource** – The Process Platform OTDS resource ID.
4. Click **Save**.
5. Restart the OpenText Process Suite Platform service for the organization.

Adding a Mobile App building block

To add a Mobile App building block:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Mobile App**.
3. In the properties pane, provide a **Display Name** and **Name** for the mobile application.
The Display Name is used as the default name of the mobile application.
4. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

5. Click  (Save).

The following properties are available in the mobile application.

Property	Description
Gateway URL	A read-only property that is configured within the Platform Properties. See Configuring Appworks Gateway per instance .
Gateway Status	The current status of the gateway. <ul style="list-style-type: none">• Connecting - Connection to the gateway is being attempted.• Not Reachable - The gateway is not reachable.• Not configured - The gateway is not configured.

Property	Description
	<ul style="list-style-type: none"> • Unauthorized - Unable to sign into the gateway. • Ready - The gateway is available.
List	A drop-down list of all lists within the entity.
Layout	A drop-down list of all layouts within the entity. The layout specified here uses the first form within that layout to display the details.
Allow entity instance creation	When selected, this enables creating an entity within the mobile app.
Choose App icon	Button to upload an icon for the mobile app.
Current status	<p>The current status of the mobile app. The following values are available:</p> <ul style="list-style-type: none"> • Unknown - The gateway is not accessible or it is loading. • Undeployed - The mobile app is not currently deployed and can be scheduled for deployment • Scheduled - The mobile app is queued within the AppWorks gateway for deployment. • Deploying – The mobile app is currently being deployed. • Deployed - The mobile app is deployed to the AppWorks gateway. • Enabled - The mobile app is enabled on the AppWorks gateway. <p>Depending on the speed of your environment, some intermediate statuses may not be displayed.</p>
Mobile App management	Buttons for managing the Mobile Application within the AppWorks Gateway. The following buttons are available: <ul style="list-style-type: none"> • Delete – Removes the mobile app from the AppWorks gateway. • Deploy – Deploys the mobile app to the AppWorks gateway. • Redeploy – Deploys the mobile app to the AppWorks gateway when it is already deployed. • Enable – Sets the mobile app to enabled. This is available only after the Mobile App is deployed. • Disable – Disables the mobile app in AppWorks. This is available only after the Mobile App is deployed.

The mobile app management option buttons are enabled or disabled based on the current status of the mobile app. For example, if the **Current Status** is **Undeployed**, only the **Deploy** option is enabled.

Current Status	Button		
	Delete – state	Deploy/Re-deploy – state	Enable/Disable – state
Unknown	Delete – disabled	Deploy – disabled	Enable – disabled
Undeployed	Delete – disabled	Deploy – enabled	Enable – disabled
Deployed	Delete – enabled	Redeploy – enabled	Enable – enabled
Enabled	Delete – enabled	Redeploy – enabled	Disable – enabled

Mobile App deployment messages

When deploying a mobile app through a CWS project, messages about the deployment are reported in the project publishing console.

Mobile app panel support

When assigning a layout to the Mobile App building block, the following panels are supported for display in the mobile app client:

- [Form](#)
- [Discussions](#)
- [Actions](#)

The Results panel is displayed, but this is not taken from the item layout. This is generated from the list selected within the Mobile App building block.

Other panels designed in the layout are not shown in the mobile app.

The name defined for the panel within the layout designer is used for each available panel. This can be used to help distinguish between different forms.

The related panel option within the layout designer is supported. This enables you to select forms and discussions from a related to-one or child-to-parent entity relationship. See [Adding panels from a related entity to a layout](#).

To enable switching between the available panels within the mobile app, each panel is listed within the AppWorks hamburger menu. Opening the hamburger menu provides the list of supported panels that can be viewed. These are displayed in the mobile app specific section of the menu. Tapping the panel within the menu closes the hamburger menu and presents the panel within the main mobile app screen.

Using Inbox lists

When you install Process Suite, a shared application called Inbox Task Management is automatically installed. The Inbox Task feature displays items from a user's Process Platform Inbox in a Process Experience list. The following lists are displayed.

- All Tasks - Shows all tasks assigned to the user or tasks that are associated with a case Worklist or role that the user has membership to.
- Personal Tasks - Shows the user's personal tasks.
- Teams Tasks - Shows the tasks that are sent to the teams that the user belongs to.
- Roles Tasks - Shows the tasks that are sent to the roles that the user belongs to.
- Worklists Tasks - Shows the tasks that are sent to the lists that the user belongs to.

A user can select a task from a list and work on it. If there are no tasks in the Inbox, these lists are shown but they do not contain any tasks.

Note: Inbox Task Management does not support Notifications used in the MyInBox Explorer.

Note: If someone changes the Inbox properties in Process Platform after the Task Management application is installed, the changes are not reflected in Process Experience.

To make the Inbox lists available in Process Experience, you need to set solution security for the Inbox Task Management application. See [Defining security for Inbox Task Management](#).

Chapter 7

Managing content

Content Management is provided by the File and Content List building blocks. The person who installs Process Suite specifies whether Content Server should be used as the document store for files that are added using these building blocks. Regardless of the document store that is selected, a Choose File option is available when Process Experience users create an item. They can use this option to upload a local file and display it in Process Experience.

Note: For information about installing and configuring Content Server, OpenText Directory Services (OTDS), and other related products, see the documentation for those products, the [Process Suite Installation Guide](#), and the [Process Suite 16 Supported Environments](#).

If Content Server was selected as the document store

Process Experience users have several options for checking out, checking in, and working with documents in Content Server. For example, they can search for a file, check it out to make changes, and then check it in. They can also Link or Copy the document to a selected item in the Results list.

If Content Server is used, the following options are available for working with files.

Note: Files that are linked to or referred to by an entity cannot be moved or deleted, as they may be linked to or referred to by other entities.

Option	Description
Open	Opens the file using the application in which it was created.
Rename	Renames the file to a name that you specify.
Download	Downloads the file to your computer.
Upload	Enables the user to upload the file from the local file system.
Search	Enables the user to look up files in the DMS and Link or Copy a selected file as an attachment to the entity. Search is available when no files are already added to the Content List.
Move	Moves a document between groups. The location of the file in DMS does not change.
Delete	Deletes the selected file if it was uploaded to the entity and if it is not referenced by any other entity. The file is not deleted until all the

Option	Description
	references to it are deleted.
Metadata	Displays information about the file (such as Date Created, Date Modified, and so forth) but does not display the file content.
Audit	Records a history of all changes made to the file, along with information such as the user who made the change and when the change was made.
Check Out	Checks out the file so that you can make changes.
Check In	Checks in your changes as a new version.
Versions	Lists all the revisions to the file. Any change to the file is a revision and is versioned.
Open in Viewer	Displays the file content in the Brava viewer.

If Content Server was not selected as the document store

Content is saved using the document store configured through the Document Store API. The content is stored in a folder named for the application, which is formed from the company and project names entered in Collaborative Workspace) and, below that, in a sub-folder with the Folder name specified in the File properties.

If no folder was specified, a separate unique folder is generated for each content-capable entity. The "out of the box" default document repository configuration uses XDS. The document store is configured through the System Resource Manager within the Process Platform Explorer by selecting the Repository service container and clicking the Document Store tab, which provides options additional to XDS.

Adding files

Use the File building block to enable users to add a file to an item and display it in Process Experience. For example, if an Insurance Claim entity contains the File building block, a Process Experience user can include a document such as a police report or a map of an accident location to an item that is based on that entity.

Tip: You can add only a single File building block to an entity. This enables Process Experience users to add a single file to each item based on that entity. To enable Process Experience users to add multiple files to each item based on the entity, see [Adding content](#).

To add a File building block to an entity.

1. In **Workspace Documents**, open the entity.
2. Click **Add > File**.

Note: The Name is supplied and cannot be changed.

3. In Folder, type an optional folder name. The folder specifies a location within the document storage area for documents belonging to this entity.
4. In Max file size (in bytes), type the maximum size of the file that can be uploaded.

In Process Experience, if the upload file size is greater than the specified size, a message is displayed and the **OK** button is disabled.

5. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

6. Click  (Save).

Note: Files that are linked to or referred to by the entity cannot be moved, as they may be linked to or referred to by other entities.

Uploading and downloading files in Process Experience

When a user uploads a file, the file is copied from the user's local system to the server. There is no relation, connection, or pointer to the original source file after the upload is completed. The uploaded content is completely separate from the original source file, which can be deleted, edited, moved, and so forth.

In order to edit the file on the server, the file must be downloaded to the user's local system, which makes a third copy of the file. After editing the file on the local system, (it could be the original copy, it could be the downloaded copy, or it could be a completely different file entirely) the user can then upload that edited or new file and overwrite the copy that was previously uploaded to the server.

A file that is uploaded from a user's local system to the server overwrites the existing file on the server. Therefore, if a user repeatedly edits the local copy and then uploads the edited version, users who download that document will get the edited version. If another user has downloaded the file from the server in the meantime, that user's local copy is not updated with the recently uploaded changes.

If the file on the server is deleted, the user's local copy is not affected. If a user deletes the local copy, the file on the server is not affected. If a user deletes an item with uploaded content, the corresponding content is also deleted from the user's local system.

Uploading a file in Process Experience

When a user creates an item for an entity that includes the File building block, the Create form contains a **Choose File** option for uploading file content.

When viewing an item that includes a panel for action buttons, the default action buttons include **Upload** and **Download** buttons for a user to upload and download content. If the workitem does not contain content, the **Download** button is disabled.

A user can display content in a Web Content panel using the following URL:

{system.baseURL}app/entityRestService/Items({item.\$meta.\$itemId})/ContentStream

Where "{system.baseURL}" and "{item.\$meta.\$itemId}" are resolved to the appropriate base URL and item ID, respectively.

Adding content

Using the Content building block, you can enable Process Experience users to add files to a selected item and perform document management operations on a given file. The supported actions for a specific file depend on the underlying document store configuration. For example, if the document store connector is configured with Content Server, application users see actions such as upload, download, check in, check out and so forth. The set of supported actions may be different when the document store connector is configured with OpenText Core.

Unlike the File building block, which enables users to add only a single file to an item, the Content building block enables users to add multiple files to an item. You also have the option to specify the maximum number of files to be uploaded. For example, a user who selects a job from a list can attach a candidate's resume, offer letter, and so forth.

To edit an attachment, the user downloads it, makes changes, and uploads the edited version, overwriting the previous version.

You can add only a single Content building block to an entity. The Content building block automatically creates a child entity that contains the components required to manage the content list. The child entity includes the Identity, File, Title, Content Template, and Layout building blocks.

Caution: Do not delete any of the building blocks that are automatically created in the child entity and modify them with caution. You can, however, create additional building blocks such as lists. You can also edit the Content Template building block if you want to create your own folder hierarchy into which documents will be stored in Process Experience.

To configure a Content building block:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Content**.

Note: The name is supplied and cannot be changed.

3. In **Root path**, accept the default path for the content or type a new one.

The default value is

`${system.organization}/${solution.name}/${contentbuildingblock.path}`

At runtime, the root path value for this expression is resolved to

`<OrganizationName>/<PackageName>/<EntityName>/<IdentityID>/contents`

A sample evaluated value in this case will be as follows:

`MyOrganization/MyCompanySampleProject/Employee/1/contents/`

The root path can be a combination of an expression and a hard-coded value. For example, the root path provided as

`${system.organization}/MySolution/${item.Properties.EmployeeName}`

is resolved at runtime as

MyOrganization/MySolution/RichBanker

4. In **Maximum number of files that can be uploaded**, type the maximum number of files that a user can upload. The value must be greater than or equal to two. The default value is **No Limit** (the user can upload any number of files). If you specify a value, the user can upload any number of files up to the specified maximum.

5. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

6. In the Content Details pane, click **Configure Child Entity**.
7. Click **Default Template > Default Template** to see the default Content Template. The Default folder is displayed under **Folder Structure**. If necessary, you can add siblings and child folders as required. When you point to the folder name, four icons are shown:

To create a new sibling folder, click .

To add a child folder click .

To delete the current folder, click .

To rename a folder, click .

8. Click  (Save) and close the Default Template window.

9. Select **File**.

In Max file size (in bytes), type the maximum size of the file that can be uploaded or checked in. In Process Experience, if the upload file size is greater than the specified size, a message is displayed and the **OK** button is disabled.

Note: It is recommended that you also include a Form panel so that users can see the item, its attachments, and a preview of a selected item.

10. Click  (Save).

To enable the display of the content list in Process Experience:

1. Open the parent entity.
2. In the list of building blocks, select **Layout** and create a layout that contains at least a Contents panel and a Preview panel. See [Adding layouts](#).

Tip: It is recommended that you also include a Form panel so that users can see the item, its attachments, and a preview of a selected item.

Example

In the following example, the default layout for the Company entity contains the following panels:

- The Form panel displays information about a selected company.
- The Content panel displays the folder structure created using the Content Template building block.
- The Preview panel displays the selected attachment.

If a user opens multiple attachments, each one appears as a separate tab in the Preview panel.

Adding a business workspace

The OpenText Extended ECM platform extends the transactional process management capabilities of leading business applications with comprehensive Enterprise Content Management (ECM) capabilities, including document management, records management, and collaboration. This is achieved by bringing application context (such as business data) and content together using business workspace and business object concepts of the extended ECM platform.

A business workspace is a container in Content Server that holds the complete information (such as metadata, documents, images, and other objects) that are relevant to a business object. Using the Business Workspace building block, you can enable an entity with extended ECM capabilities. This helps application users to manage business workspaces and content in the application through various widgets along with entity data.

When you add a Business Workspace building block to an entity, the following panels are available for configuration in the layout:

- The Business Workspace panel enables Process Experience users to perform the Content Server document management operations (such as upload, download, copy, move, and

so forth) in the workspace,

- The Business Attachments panel enables Process Experience users to add existing content in Content Server to the entity.

To add a Business Workspace building block:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Business Workspace**.
3. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

4. With the Business Workspace building block selected, click **Configure**.
The **Available Properties** list opens, displaying a list of all properties and building blocks added to the entity.
5. Select one or more properties to store in Content Server as metadata of the workspace.
6. Click  (Save) and close the window.

To configure a Business Workspace panel and a Business Attachments panel in a layout:

1. Open the entity that contains the Business Workspace building block.
2. Create a layout that contains Action Bar, Form, and Business Workspace panels. See [Adding layouts](#).
 - The Form panel displays items to users.
 - The Actions panel shows actions (such as Synchronize and Open Workspace) required for Extended ECM integration.
 - The Business Workspace panel enables users to manage the workspace and content in the workspace.
3. Drag the Business Attachments panel to the middle of the Business Workspace panel. This will be shown as a tabbed panel in Process Experience.
4. Click  (Save).

To configure an action bar that shows actions related to the business workspace:

1. Open the entity that contains the Business Workspace building block.
2. Create an action bar that contains the Open, Delete, Synchronize Workspace, and Open Workspace actions. See [Creating an action bar](#).

Tip: The order of the buttons is determined by the order in which you select them.

3. Click  (Save).

To configure the Security building block:

1. Open the Security building block for the entity that contains the Business Workspace building block.
2. In Business Workspace, select some or all of the following options based on your requirements:
 - **Synchronize Workspace** – Creates a workspace when an item is created and pushes entity data (based on the properties selected in the Business workspace building block) as metadata of the workspace.
 - **Open Workspace** - Opens the Content Server business workspace in a popup window.
 - **Add business attachment** – Adds a link to another document present in Content Server.
 - **Delete business attachment** – Deletes existing business attachments.
 - **Open in Brava Viewer** - View attachments using Brava Viewer.
3. Click  (Save).

Chapter 8

Facilitating collaboration

The Discussion building block provides users with a message board that facilitates communication about items.

The Email building block enables users to Send, Receive, Reply, and Forward emails along with attachments from Process Experience.

Adding a message board

You can use the Discussion building block to add a message board to an entity. This facilitates collaboration among Process Experience users working on a particular item. The Discussion building block creates a child entity that contains a list of topics and replies. The child entity includes the Identity, DiscussionNote, and Display Organization building blocks. To display the message board containing the topics and replies, create an item layout in the parent entity that contains the Discussions panel.

The Discussion building block makes the **Add comment** action available for adding to an action bar. If added, users can enter and view added comments. See [Creating an action bar](#).

You can specify the security for adding comments in the Security Editor. See [Configuring security](#).

You can add only one Discussion building block to an entity. After you add the Discussion building block, it no longer appears in the list of building blocks that are available for adding.

Note: The Discussion building block and the Discussion Panel cannot be used in a child entity.

To create a message board:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Discussion**.

Note: The Name is supplied and cannot be changed.

3. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

4. Click  (Save).
5. Create a Full Layout for the entity and include a Discussion panel and a Form panel. The Form panel should include a form (such as the Default form) so that you can refer to the item information in the discussion.

See the *Process Experience User's Guide* for information about how to use a message board in Process Experience.

Providing email functionality

Using the Email building block and the Email template building block you can provide email functionality in your applications. This enables Process Experience users to send, receive, reply to, and forward emails (and attachments) from a selected item. The files that are to be attached to the outbound mails must be added to the item through the Content building block. Files received through inbound emails are stored in the Content building block that is provided as part of the Email building block. Inbound and outbound email mailboxes must be configured in the Process Experience Administration page.

Note: If you expect Process Experience users to include attachments in email, you must include a Content building block in the entity and a Content panel to the layout. Users will upload files that they intend to use as attachments to the Content panel and select them from there. See [Adding content](#), [Adding layouts](#), and [Adding panels to a layout](#).

The overall sequence for enabling email capabilities follows:

1. Configure the email server that is required for the application. This is a one-time system wide task. See [Configuring the E-mail connector](#).
2. Add an Email building block.
3. Add one or more Email templates. See [Creating an email template](#).
4. Add an email panel to a layout. See [Adding layouts](#).
5. Publish your application. See [Publishing a project](#).

When you add an Email building block, a child entity that will contain the list of files is automatically created. The child entity includes Content, Email Log, and Identity building blocks.

Building block	Description
Content	Used to show content that will be used in emails (such as documents and attachments) in Process Experience. The Content building block includes the following building blocks: <ul style="list-style-type: none">• Content Template• File• Identity• Layouts - contains a Default layout

Building block	Description
	<ul style="list-style-type: none"> Title
Email Log	Maintains an audit trail of all received (inbound) and sent (outbound) emails. It is not configurable and system defined.
Identity	Used to include the primary key properties of the entity so that they can be used in a list to search for an item by primary key. No configuration is required.

Caution: Do not delete any of the building blocks that are automatically created in the child entity and modify them with caution.

If an Email building block is added to an entity, a **Send Email** option is automatically added to the action bar for an item that is opened in Process Experience.

The Email building block provides an option for saving the attachments of received email to the Content building block. This enables users to see email attachments at a single location in the Content panel and to easily manage them using Content panel features. See [Adding content](#).

To add and configure the Email building block:

1. In **Workspace Documents**, open the entity.
 2. Click **Add > Email**.
 3. Click  (Search).
The Select Email Configuration dialog box opens.
 4. If the email configuration already exists, select it and click **OK**.
- Note:** An Email Configuration can be bound to multiple entities by using the Email Building Block within a workspace. Use the Process Experience Administration tool to specify email configuration parameters such as From, Display Name, Reply to, and Email profile User ID. See [Specifying the email configuration](#).
5. If you need to create a new email configuration:
 - a. Click **New > Email configuration**.
 - b. Click  (Save).
 - c. When prompted, enter a name, description, and location for the email configuration and click **OK**.
 - d. Close the window, select the email configuration, and click **OK**.
 6. If you want to enable saving attachments of incoming email in a specified folder in the Content panel:
 - a. Select **Save the attachments of incoming e-mail to the selected content folder**.

The Content location list displays the folder paths that were created in the Content

building block.

- b. Select a content location and click **OK**.

Note: In Process Experience, the attachments of incoming email that are saved to the selected folder are listed in the Content panel.

7. Click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

8. Click **Configure child entity**.

The Child entity contains Content, EmailLog, and Identity building blocks. Attachments to the e-mail will be stored and managed using the Content building block. See [Adding content](#).

9. Click  (Save).

Creating an email template

Use the Email template building block to create the templates that will be available for Process Experience users to send emails. Users can also select Blank to create an e-mail that does not use a template.

If you plan to include attachments in a template, you must first upload them using the Content building block.

Before you begin:

- Add the Email building block to the entity. See [Providing email functionality](#).

To add an e-mail template:

1. In **Workspace Documents**, open the entity.
2. Click **Add > Email template**.
3. Type the Name of the template and click **Add**.

Tip: If you want to immediately create additional building blocks, click **Add and continue**. You can configure them later by selecting each one from the list of building blocks.

4. Click **Configure**.

5. Specify some or all of the following values for the email template:

- To - The email addresses of the email recipients.
- Cc - The email addresses to copy on the email.
- Subject - The subject of the email.
- Body - The text of the email. Various formatting options are available.

You can use expressions (see [Expression language](#)) and text patterns (see [Text patterns](#)) that will be evaluated in Process Experience in the To, Cc, Subject, and Body.

For example, you can use the following text pattern in an email template:

```
{item.Properties.propertyname}
```

Note: To retain the relationship between an email and an item, users of your application should not change the subject of the email while replying.

Configuring the E-mail connector

The E-mail connector enables users to send and receive emails. It supports single mail servers for both incoming and outgoing emails. Users can send simple emails with the `SendMail` SOAP request using the operation name `SendMailoperation` and construct more complex emails with the `SendMailMIME` SOAP request using the operation name `SendMailMIME`. You can specify email attachments with either operation.

You can retrieve email on demand with the `GetMail` or `GetMails` SOAP requests using the corresponding operation names `GetMailoperation` or `GetMailsoperation`. You can also retrieve emails automatically using the mailbox configuration that defines the recipient's email address to monitor and the action to perform when an email is received for the specified recipient. You can retrieve email attachments with either on demand or automated retrieval of emails.

The E-mail connector is automatically deployed when Process Platform is installed. This topic covers configuring the 3.x series of the E-mail connector.

Before you begin:

- Ensure that the E-mail connector application package was deployed during the Process Platform installation.
- Ensure that you have an MS-SQL Server, Oracle, MySQL, or PostgreSQL database to hold the processed emails. If you are using a MySQL database, you must use version 5.5.14 or later.

Creating a database configuration

Note: You need to create a database configuration only if you do not intend to use the Process Platform System database with the E-mail connector.

You can choose to create the database configuration either before or after the E-mail connector configuration. This section explains how to create the database configuration before configuring the E-mail connector.

To create a database configuration before configuring the E-mail connector:

1. On the Welcome page, click  (System Resource Manager).
The System Resource Manager window opens and displays the service containers.
2. On the toolbar, click  (Manage Database Configurations).
The Manage Database Configurations page opens.

3. Click  (Insert).
A new row is appended to the table and a section opens where you can enter the database configuration details.
4. For Name and Description, enter the name and description of the database configuration.
The description is optional.
5. Enter the relevant details in the corresponding fields.
For information about the fields, see [JDBC Details Interface](#).

Notes:

- To create a database, select **Create New Database** and enter the required details for DBA Name and DBA Password.
- You can create a tablespace while creating a database (for PostgreSQL) or new user (for Oracle) if you select **Oracle Thin/OCI** or **PostgreSQL** from the JDBC driver list. See [Creating a Tablespace](#).

6. Click  (Test Connectivity) at the top of the page to test the connection.
7. Click  (Save).

A database configuration is created and the name, description, and organization (under which the database configuration is created) are displayed in the new row.

Note: For a new user in Oracle, only Create permissions are granted to the user; for other permissions, you must connect to the Oracle server using the client and select the permissions explicitly.

Configuring the E-mail connector

Configuring the E-mail connector involves the following configurations:

- Inbound
- Outbound
- Mailboxes
- Database
- Key managers

To configure the E-mail connector:

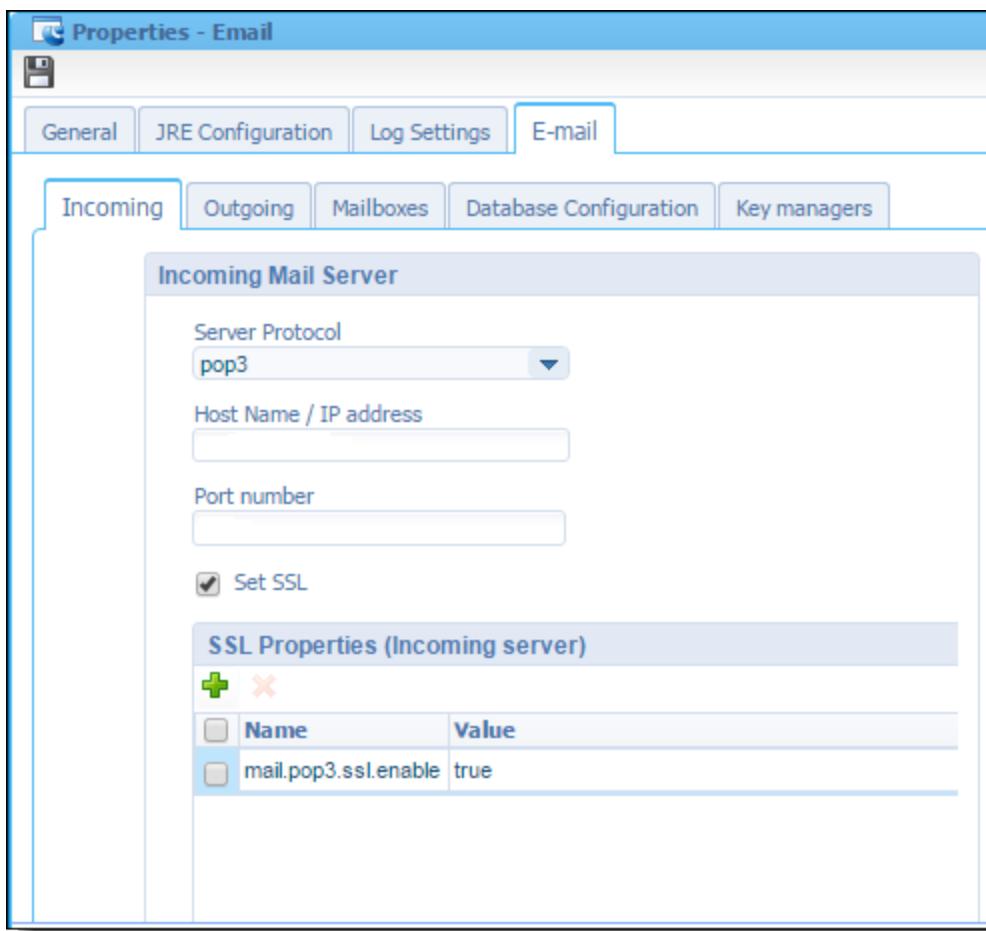
1. On the Process Platform Welcome page, click  (System Resource Manager).
The Email service container is displayed in the Email service group. These are created automatically when Process Platform is installed.
2. In the Service Containers pane, right-click the E-mail service container, and then click **Properties**.
3. To automatically restart the container, on the General tab, in the Startup Type list, select **Automatic**.

4. Click the E-mail tab.

A number of tabs are displayed.

5. On the Incoming tab, enter information for the Incoming Mail Server.

All the values are specific to the email server instance set up in your organization. Contact your IT department for the required values.

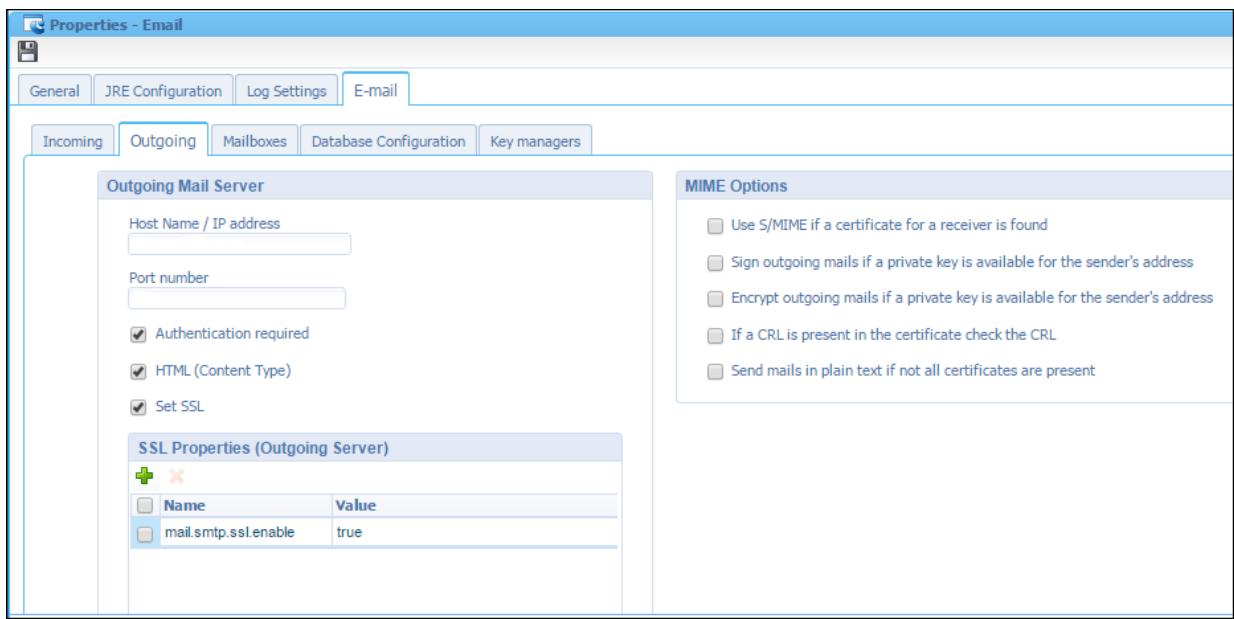


The following table describes each option.

Option	Description
Server Protocol	Network protocol to transfer data. Options available are pop3 and imap. The POP3 protocol only supports retrieval of email from the default mailbox, INBOX while IMAP supports email retrieval from any mailbox configured for the email account that is specified in the email retrieval request.
Host Name / IP address	IP address of the incoming mail server.

Option	Description
Port number	Port number of the incoming mail server.
Set SSL	Whether to use an encrypted connection.
SSL Properties (Incoming Server)	Name and value of the incoming server. The required values are populated when Set SSL is selected. Your IT department will inform you if more SSL properties are required.

6. On the Outgoing tab, enter the following information for the Outgoing Mail Server. All the values are specific to the email server instance set up in your organization. Contact your IT department for the required values.



The following table describes each option.

Option	Description
Host Name / IP address	IP address of the outgoing mail server.
Port number	Port number of the outgoing mail server.
Authentication required	Whether authentication is required.
HTML (Content Type)	Whether HTML content type is set in the outgoing email header.
Set SSL	Whether to use an encrypted connection.
SSL Properties (Outgoing Server)	Name and value of the outgoing server. The required values are populated when Set SSL is selected. Your IT department will inform you if more SSL properties are required.

The Outgoing tab also provides the following MIME options. These options are not mandatory.

Option	Description
Use S/MIME if a certificate for a receiver is found	Whether S/MIME is enabled.
Sign outgoing mails if a private key is available for the sender's address	Whether to sign an outgoing mail if a private key is available for the sender's address. Usually, this only works for a system email address and not for individual users because their private keys are not accessible.
Encrypt outgoing mails if a private key is available for the sender's address	Whether to encrypt mails. Mails can be encrypted when the public key of a user (which is usually stored in Active Directory) is accessible. Encryption ensures that only that user can read the mail.
If a CRL is present in the certificate check the CRL	Whether to check if the CRL (Certificate Revocation List) is present for a certificate.
Send mails in plain text if not all certificates are present	Whether to allow sending of non-encrypted mails if S/MIME is enabled.

- On the Mailboxes tab, enter the following information. This is required for the E-mail connector to perform automatic polling for incoming mail.

Option	Description
Number of poller threads	<p>Number of threads to use when polling for incoming email. Each thread is a separate process. You can increase the value for more emails to be processed concurrently.</p> <p>Note: If you are specifying multiple mailbox configurations, you can increase the number of poller threads. However, a large number of threads can negatively impact the overall performance of your Process Platform installation.</p>
Email box configuration	XML you enter against the schema when the connector is started. The connector validates this XML. Note: If automatic polling for incoming emails is not desired, the default mailbox configuration is valid and need not be modified.

A sample configuration XML for the Email box configuration follows.

```
<emailboxes xmlns="http://schemas.cordys.com/1.0/email/configuration">
    <emailbox>
        <name>SolutionStore_Ticketss</name>
        <username>test123@gmail.com</username>
        <password>test123</password>
        <folders>
            <folder>INBOX</folder>
        </folders>
        <triggers>
            <trigger appliesTo="INBOX">
                <name>AllFolders</name>
                <rules>
                    <rule section="SUBJECT">
                        <pattern type="REGEX">
                            <value>.+</value>
                        <store>
                            <token>
                                <name>Subject</name>
                                <value>0</value>
                            </token>
                        </store>
                    </pattern>
                </rule>
                <rule section="TO">
                    <pattern type="REGEX">
                        <value>.+</value>
                    <store>
                        <token>
                            <name>To</name>
                            <value>0</value>
                        </token>
                    </store>
                </pattern>
            </rule>
            <rule section="FROM">
                <pattern type="REGEX">
                    <value>.+</value>
                <store>
                    <token>
                        <name>From</name>
                        <value>0</value>
                    </token>
                </store>
            </pattern>
        </rule>
        <rule section="MULTIPART">
            <pattern type="CUSTOM">
                <value>com.eibus.applicationconnector.email.sample.Base64Plugin</value>
            <store>
```

```

<token>
    <name>Base64Attachment</name>
    <value>string</value>
</token>
</store>
</pattern>
</rule>
</rules>
<message>
    <method>receiveEmail</method>
    <namespace>http://schemas.cordys.com/entity/email/1.0</namespace>
    <user>cn=cordys,cn=organizational
users,o=system,cn=cordys,cn=defaultInst,o=vanenburg.com</user>
    <sync>true</sync>
    <namespacemappings>
        <namespacemapping>
            <prefix>ns</prefix>

<namespace>http://schemas.cordys.com/entity/email/1.0</namespace>
            </namespacemapping>
        </namespacemappings>
        <input xmlns:ns="http://schemas.cordys.com/entity/email/1.0">
            <ns:from />
            <ns:subject />
            <ns:body />
        </input>
        <mappings
xmlns:tns="http://schemas.cordys.com/1.0/email/configuration">
            <mapping>
                <source>./ns:from</source>
                <value src="From" />
            </mapping>
            <mapping>
                <source>./ns:subject</source>
                <value src="Subject" />
            </mapping>
            <mapping>
                <source>./ns:body</source>
                <value src="Base64Attachment" />
            </mapping>
        </mappings>
    </message>
</trigger>
</triggers>
<automaticrestart
xmlns:tns="http://schemas.cordys.com/1.0/email">true</automaticrestart>
    <restartinprogress
xmlns:tns="http://schemas.cordys.com/1.0/email/configuration">false</restartinpro
gress>
    <emailbox>
</emailboxes>
```

Customize the following parameters in the XML configuration for your email account:

Parameter	Set to
<username>test123@gmail.com</username>	User ID of your email account.
<password>test123</password>	Password of your email account.
<user>cn=cordys,cn=organizational users,o=system,cn=cordys, cn=defaultInst, o=vanenburg.com</user>	User DN for the triggered process specified in the <method> tag. Replace the following: In o=system, replace system with your organization name. In cn=defaultInst, replace defaultInst with the instance name under which Process Platform is installed on the server. In o=vanenburg.com, replace vanenburg.com with the domain name.

9. On the Database Configuration tab, enter the following information.

The screenshot shows the 'Properties - Email' dialog box with the 'Database Configuration' tab selected. The configuration details are as follows:

- Storage provider:** Database
- Database details:**
 - Compress data:
 - Number of execution threads: 1
 - Max messages per DB poll:
 - Polling interval for executing the processing of matching emails: 30000
- Organization:** system
- Select Database Configuration:** Cordys System
- Driver:** JDBC
- Default Database:** defaultInst
- DB User:** sa
- JDBC Driver:** Microsoft SQL Server
- Connection String:** jdbc:sqlserver://localhost:1433;sendStringParametersAsUnicode=false
- JDBC Driver Class:** com.microsoft.sqlserver.jdbc.SQLServerDriver
- JDBC Driver XA Class:** com.microsoft.sqlserver.jdbc.SQLServerXADataSource

Option	Description
Storage provider	Storage provider for the connector. Select Database. Note: To test the connector, select the In Memory (not for production) storage provider because you do not require persistence. Ensure that you do not use this storage provider in any production environment because you might lose emails.
Max nr of rows	Maximum number of rows to use in the database.
Compress data	Whether the E-mail connector must zip the data before it is written to the database. This is very useful in limiting the storage.
Number of execution threads	Number of threads to use for the execution engine, that is the number of SOAP requests that are sent in parallel.
Max messages per DB poll	Number of messages to retrieve from the database when the execution engine looks for new work.
Polling interval for executing the processing of matching emails	Interval to poll the database for emails to process.

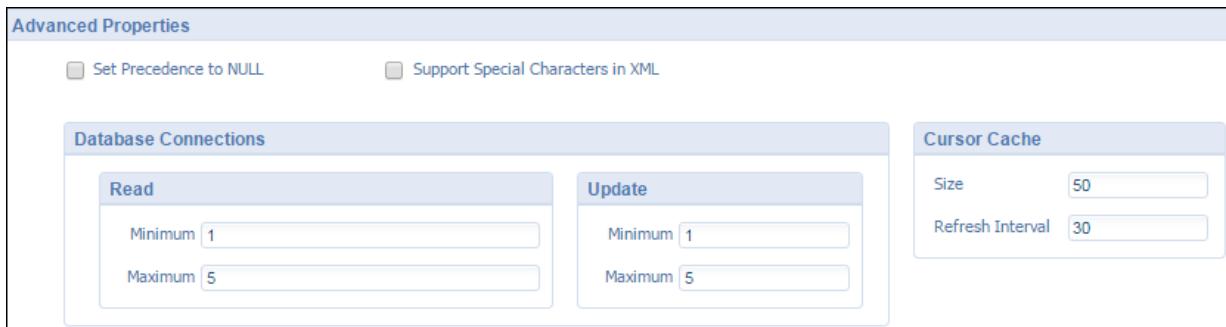
10. On the Database Configuration tab, perform the required steps based on the database you are using:

Database	Steps
Cordys System database	<ul style="list-style-type: none"> a. In the Select Database Configuration list, select Cordys System. All the fields are filled automatically. b. Click Save. c. Select the Restart Service Container check box, and then click Yes. d. Click Refresh. All the service containers are refreshed.
Another database that you created before beginning the database configuration	<ul style="list-style-type: none"> a. In the Select Database Configuration list, select the database configuration you created before you began configuring the E-mail connector. All the fields are filled automatically. Note: You might have created a database in the previous section, Creating a database configuration. b. Click Save.

Database	Steps
	<p>c. Select the Restart Service Container check box, and then click Yes.</p> <p>d. Click Refresh. All the service containers are refreshed.</p>
Another database that you are yet to create	<p>If you have not created a database configuration before configuring the E-mail connector, you can create it here.</p> <p>a. In the Select Database Configuration list, select New Database Configuration. The New Database Configuration window opens.</p> <p>b. For Name and Description, enter the name and description of the database configuration. The description is optional.</p> <p>c. Enter the relevant details in the corresponding fields. For information about the fields, see JDBC Details Interface.</p> <p>Note: To create a database, select Create New Database and enter the required details for DBA Name and DBA Password.</p>

Under **Advanced Properties**, enter the required information related to setting precedence to NULL, supporting special characters in XML, database connections, and cursor cache.

Note: When using an Oracle database, you must select the Support Special Characters in XML check box.



- On the Key Managers tab, enter the key manager configuration, if necessary:

```
<keymanagers xmlns="http://schemas.cordys.com/1.0/email/configuration" >
</keymanagers>
```

Note: The application package contains a key manager called com.eibus.applicationconnector.email.sample.GenLDAPConnectorKeyManager. This is the key manager to use with the Generic LDAP Connector for certificate lookup.

12. Click  (Save).

Invoking the SetProfile Web service

During the E-mail connector configuration, if you selected the **Authentication required** check box on the Outgoing tab, the SetProfile Web service will need to be invoked at least once prior to executing GetMail/GetMails (incoming) and/or SendMail/SendMailMIME (outgoing) to provide the mail account credentials.

This does not apply to mail received by the polling mechanism as configured on the Mailboxes tab since the mail account credentials are included in the <emailbox> configuration. It is not expected that an end user will need to invoke the SetProfile Web service – the service that is invoking GetMail/SendMail should be obtaining the user credentials and invoking SetProfile as needed – but it can be done with the following steps:

To invoke the SetProfile Web service:

1. On the Welcome page, click **Web Service Interface Explorer**.
The Web Service Interface Explorer window opens and displays the search options.
 2. In Keyword, enter **setprofile** and click **Find**.
The search results are displayed.
 3. Select the SetProfile Web service operation (tagged with the Email service group and Email implementation type), right-click, and then select **Test**.
The Operation Test Tool opens and displays the SOAP request.
 4. In the SOAP request, replace the following parameters with the values required for the email account to be used for subsequent invocations of GetMail/SendMail:
 - **displayName** (optional)
 - **mailId** (use the email address from which you want to send/receive mail)
 - **password** (use the password for the email address from which you want to send/receive mail)
 - **userId** (use the userid for the email address from which you want to send/receive mail. This is normally the email address minus the '@' and anything after it)
5. Click **Invoke**.

Chapter 9

Sharing and re-using applications

You can save considerable development time by using existing applications as the basis for creating new applications.

For example, assume that someone created a Human Resources application that contains an entity called Employee with a property called Office Location and designated both the entity and the property as available for use in other applications. If you create a Facilities Management application, you then can use the Office Location property of the Employee entity in that application.

Customizing an application

You can create a completely new application by customizing an existing application to your specific requirements. For example, you can customize one of the applications that OpenText provides. You do this by importing the out-of-the-box version of the application and then customizing it to your organization's specific requirements. When you customize an application, you can replace the imported entities and building blocks with ones that are suitable for your organization's particular requirements. The application that you imported remains intact and your changes are used to create a new application. When you customize an entity containing a child entity, you can also customize the child entity.

Assume that you import a generic Contract Management application package that provides common information about all types of contracts. It contains an entity called Contract with properties such as Contract Number, Client, Address, Number, Phone Number, and so forth. However, your organization needs an application that is customized specifically for Sales contracts so you need to customize the generic Contract Management application so that you can add other properties to it. For example, you may need to add a property called Territory and add it to the forms used to create and view contracts.

Using customization, you can also share applications within your organization across business units. For example, if the IT team created an application specific for their IT Services, the Cloud Services or Professional Services business unit could import the IT application and then customize it according to their business needs.

Configuring the custom application

After you [import an entity](#), right-click the entity, and click **Customize**, you can configure the entities and building blocks that are appropriate for your application by adding,

renaming, and replacing certain types of building blocks. When you open the entity, building blocks that cannot be added, replaced, or renamed are disabled.

Note: The replacement name of an entity or building block in a custom entity must be the same as its name in the original entity. You are not prevented from changing it, but validation of the entity will fail if the names are different.

Following are some considerations for configuring the custom application:

Customizing the domain model - Generic applications are meant to cover the common needs of a business domain, whereas every organization in that domain has its own special needs, both in terms of the data that is stored in the application and how the application behaves under specific circumstances. In addition, business users want the application data to be presented in a way that is easy to use.

Customizing the way data is presented - After new properties are added, forms are needed where they can be given a value. In addition, end users may be expecting this new property to be shown in some of the existing lists or new lists may need to be designed. Since properties may need to be positioned on a form in different way, it does not always make sense to include the generic Create form as a sub-form on the new application. However, the original entity may contain other forms that would be useful to include as sub-forms.

Customizing the business logic - The custom application typically needs to customize the business logic from the original application or to add new business logic for managing the data. You do this by adding building blocks such as Rules, Action bars, Title, History, File, Discussion, Lists, Layouts, and others. You may also need to create new entities. See [Populating entities with building blocks](#).

Important: There is a difference between customizing an entity and [Advanced domain modeling using subtyping](#). The main difference is that with customization the objective is to change the behavior of an existing entity, whereas by subtyping an entity you are creating a new entity that reuses what was in the existing entity. For example, when you customize an entity called Account, the resulting entity is still called Account. When you subtype the Account entity, the resulting entity is a special type of entity (such as EUR Account or CAN Account). With subtyping, you can distinguish between the original and the subtype - the original holds generic elements and the subtype its unique elements. With customization, you see only one Account entity.

Upgrading the base application

One of the most important aspects when customizing applications is the ability to seamlessly upgrade to newer versions of the base application. Using Entity Modeling, if issues are fixed or other changes are performed upgrading to the version that contains the fixes does not require reengineering of the customizations. This is also true when you upgrade to newer versions of the base application in which new features are added.

Under specific circumstances, changes in a newer version of the original application may conflict with changes made in the custom application. In these cases, the system brings the conflicts to your attention and provides an easy method to resolve them.

For example, assume that you add Security to your custom version of the Account entity and then you upgrade the original application containing the original Account entity. After the upgrade, it appears that Security was added to the original Account, resulting in two Security building blocks being visible. Since this is not a valid situation, the system cannot validate your project. One way to resolve this is to select the Security you configured in your custom Account entity and then click **Replace Original**. When you do this, the Security configuration from the original Account entity disappears and your project can be validated.

Updating security

After customizing the functionality of an application by adding properties, relationships, or other building blocks, you need to define security for this functionality by customizing the Security building block of the entity. See [Defining security for a customization or subtype](#)

Generating an application package for customization

The steps to create the model package of an application are described in [Setting Properties of an Application Package](#) in the Process Platform product documentation.

Customizing Platform Packages

Process Platform also provides entities that can be used like Model contracts or Model packages. These packages are already installed and imported during Process Platform installation, and are available for all organizations. The packages are made visible in CWS by selecting the **Show Platform Packages** option from the context menu on Imported Application Packages. Their content can be used like any other imported content.

Importing entities from other applications

You can streamline application development by using entities and building blocks that were already created in another application. These entities have to be made available by their developer in an application package.

The application package can be one of the following:

- A **.cap** application package. A .cap package contains runtime information, but can also contain some designated model information. This enables creating relationships to designated entities and, via these relationships, you can use the designated entity properties and relationships in forms and other building blocks.
- An **.mpk** application package. An .mpk package contains model information. This enables customizing entities in the package by adding or replacing building blocks.

If an entity is re-used via a .cap application package, and you discover that there is a need for customization, the .cap package can be upgraded with the .mpk package of the same application version.

Note: Unless it is necessary, no explicit distinction will be made between these package types in the rest of the documentation. They will both be referred to as 'application package'.

To import entities from a .cap application package:

1. In the source application, designate the entities and building blocks that should be available for use in other applications. You designate the entities and building blocks individually. Some building blocks are not available for use in other applications.
2. Create and download a .cap application package (or update a previously downloaded package) from the source project. The entities and building blocks available for use in other applications are automatically included in the package. Each package can contain multiple entities and building blocks.
3. Send the .cap application package to the developer, generally on another Process Platform installation, who wants to reuse entities in the application package. The .cap application package has to be deployed there.
4. Import the application package in the CWS workspace. This makes the entities and building blocks that were designated as available for use by other applications available for modeling. You can create relationships only to the entities in the .cap file.

Note: When you re-use entities via a .cap application package, you must deploy the .cap application package before you can publish or package your application.

To make an entity or building block available for use in other applications:

1. In the workspace, right-click the entity and select **Properties**.
2. On the General tab, select **Available for use by others**.
3. Click **OK**.
4. In the Details pane for an existing building block, select **Available for use by others**.

Note: If **Available of use by others** is not selected for an entity, the **Available for use by others** option is disabled for its building blocks. If you select this option for an entity, you must also select it for at least one of the entity's building blocks. Some building blocks do not provide the **Available for use by others** option.

To import entities from an .mpk application package:

1. Create and download an .mpk application package (or update a previously downloaded package) from the source project.
All entities and building blocks are automatically included in the package.
2. Send the .cap and .mpk application package to the developer, generally on another Process Platform installation, who wants to reuse entities in the application package.
The .cap application package has to be deployed there.
3. Import the .mpk application package in the CWS Workspace.
This makes the entities and building blocks available for modeling. The entities in the MPK can be customized and/or subtyped for use in your application.

Note: When you are re-using entities via an .mpk application package, you must deploy the .cap application package going with it before you can publish or package your application.

Managing application packages

Application packages can be created, downloaded, deployed, imported, updated, and deleted.

To verify the package properties:

1. In the workspace, right-click the project to be packaged and select **Packaging > Package Properties**.
2. Verify that **Supported Deployment Spaces = Organization**.
3. If you want an .mpk application package, be sure that **Create Model** package is selected.

To create and download application packages:

1. Verify the package properties.
2. In **Workspace Documents**, right-click the project to be packaged and select **Packaging > Create Package**.

The packaging process starts in a separate window. The **Download Package** option is enabled if the project is validated without errors and the application package is created.

3. Click **Download Package**.

The application package is downloaded to your local computer.

Note: If **Create Model package** is selected, two application packages (.cap and .mpk) are downloaded simultaneously.

To download the latest package:

If a project is already available for download, you can download the most recent application package of that project. This option is available only if there is a package already available with the same Package File Name generated while creating a new package. In this case, there is no need to create an application package first.

- In **Workspace Documents**, right-click the project for which you want to download the most recent application package and select **Packaging > Download Latest Package**. The latest application package is downloaded to your local computer.

Note: If **Create Model package** is selected, two application packages (.cap and .mpk) are downloaded simultaneously.

To deploy the .cap application package:

1. Open the CWS Application Deployer and select the .cap application package.
2. Deploy the application package to the same organization where the application package that you want to import into is stored.

Note: If the application to deploy has package dependencies (that is, it contains references to content that resides in other projects or packages), those other packages must be deployed as well.

To import or update an application package:

Note: All entities and building blocks in an application package are imported, updated, or removed at once. You cannot import, update, or remove them individually.

1. Open the Import Application Package dialog box in one of the following ways:
 - In the toolbar for Workspace Documents, click  (Import Application Package).
 - In **Workspace Documents**, right-click **Imported Application Packages** and select **Import Application Package**.
2. Browse to the application package, select it, and click **Open**.
3. Select one of the following options:
 - To import a new application package, click **Import**.
 - To update previously imported (older or newer) application package, select it, and then click **Update if application package already exists > Import**.

Notes:

- If an imported application package has dependencies to other application packages, the imported package icon is shown with a red minus sign. The actual dependencies can be found in the imported package properties. The packages that are shown there may also have to be imported.
- Import of an application package fails if it contains a document that needs to be upgraded to the latest version and if that document has unresolved dependencies on another package. The other package can contain information needed for the upgrade and must be imported before the package with the document to be upgraded can be imported.
- The update functionality supports updating a .cap. package with an .mpk package and vice versa.

To open an entity from an imported application package:

- In **Workspace Documents**, right-click <entity> and select **Open**.

The <entity> window opens, displaying the following information:

Field	Description	Edit Rights
Name	Name of the entity	Read only
Package Name	Name of the package	Read only
Package Version	Version of the package	Read only

To view the properties of an imported application package:

- In **Workspace Documents**, right-click **Imported Application Packages** and select **Properties**.

The Application Package window opens, displaying the following information:

Field	Description	Edit Rights
Package Name	Name of the package	Read only
Package Version	Version of the package	Read only
Build Number	Build number of the package	Read only
Package dependencies	Columns Package name, Version, and Build number. Identifier of the package that contains models that are used by models in this package. Column Resolved. Indicates whether a package has been imported that contains these models. This may be a package with the specified Name, Version, and Build number, but can also be a package with the same Name but different Version and/or Build number.	Read only
Model dependencies	Columns From type, From name. Identifier of an individual model in this package. Column To type, To name. Identifier of an individual model in the selected package in the Package dependencies list that is used by the (From) model in this package. Column Resolved. Indicates that the (To) model is present in the selected package in the Package dependencies list.	Read only

To remove an imported package:

1. Right-click an imported application package and select **Delete**.
2. The used by check is performed to see if the imported application package is in use.
 - a. If the package is in use, it is not removed.
 - b. If the package is not in use, the removal is complete when the application package disappears from the Imported Application Packages section.

Note: Removing an imported application package removes all the entities it contains. The system displays an informative message if any entity is in use in another application. However, it does not prevent you from removing the application package. If you do so, all references to the removed entities are also removed and errors will occur at validation of the entities that were using data of the removed entities.

To see where an entity is used:

- In **Workspace Documents**, right-click the entity and then select **Used By**.

Testing an application that re-uses entities from other applications

When you publish an entity model application, all entities must be available in the workspace of your organization.

Therefore, when you re-use entities by importing an application package, you must take care that the .cap file of the application is deployed in the workspace of your organization before you publish.

If the .cap file is not deployed, deployed in the shared workspace, or deployed in the workspace of another organization, you will get an error when you publish the application, informing you that the re-used entities cannot be found.

Chapter 10

Using the Identity package

Open Text Directory Services (OTDS) handles user management and authentication for Process Suite. The Identity Package is a package that is automatically installed with Process Suite. The Identity Package enables you to:

- Manage organizational model master data across the tenant organization in Process Experience (runtime). Users with the appropriate privileges can create the organizational structure or model, such as organizations, organizational units, as well as worklists, business contacts, countries, and more.
- Assign users to organizations and view all users in the system.
- Establish a hierarchy for organizational units.

The Identity Package provides the following functionality:

- A complete entity modeling domain model with all of the required entities, relationships, properties, user interfaces, and so on.
- An Identity home page where users with the required privileges can use Process Experience to manage identity information, such as organizational units, subunits, and worklists.
- Usage of the OTDS push connector to synchronize user information (identities) from OTDS to Process Suite.

Notes:

- If you managed your organizational model using the Process Platform Organization Model in the past, now you will manage the model in Process Experience with the Identity Package.
- If you build entity modeling applications, it is required that you use OTDS to facilitate user management. If you have Process Platform applications built on core functionality that is not entity-based, you can continue to use User Manager to facilitate user management.

Before you begin

The Identity Package requires that you use Open Text Directory Services (OTDS). Although the Identity Package is preconfigured, you need to perform the following tasks before it can be used:

- Create the users, groups, and roles to be synchronized with Process Experience in OTDS. See the Process Platform Product Documentation.
- Make the Identity Package available in Process Experience and set solution security for the Identity Package application for both users and administrators. See [Defining security for the Identity package](#).

Using identity-related information in Entity Modeling applications

To make the OpenText Entity Identity Components visible and customizable, right-click the Imported Application Packages option and select **Show Platform Packages**.

Some entity building blocks use the Identity-related information in the package. For example, the Assignee building block accesses the Identity information in order to assign a user or role to an entity instance. The Tracking building block adds Identity-related information to the entity to track when and by whom the item was created and last modified.

As an application designer, you can also leverage the large number of packaged identity-related properties and lists by establishing a relationship to these entities. For example, you can use the Rule building block to create business logic to set purchasing spending limits by the organizational unit. You can also use the Form building block and add Identity-related properties that are available in the Identity Package to your forms. For example, you can add personal information such as First Name, Last Name, Address, User ID, and so on.

Note: You must create the relationship to the appropriate Identity-related entity in order to access or use Identity information in your existing entities (for example, via rules, forms, browse lists, and so forth).

Identity domain model

The following sections describe the entities that are provided in the Identity Package.

Some of the entities are master data entities that enable users with the appropriate permissions to add and manage the data in Process Experience. Other entities are standard entities and the data is synchronized from OTDS.

The following building blocks rely on the Identity package.

- [Assignee](#)
- [Email](#)
- [Deadline](#)

Entities

The following entities are included with the package. These entities are synchronized from OTDS. For additional information on the data that is synchronized from OTDS, see

Information pushed from OTDS to Process Platform.

- [User](#)
- [Group](#)
- [Role](#)

The following entities are also provided with the package. These entities are not synchronized through OTDS. A user with the Identity Administrator role can create them in Process Experience and view them on the Identity Homepage. Management of these identities is performed from the Identity Homepage in Process Experience.

- [Address](#)
- [Assignment](#)
- [Country](#)
- [County](#)
- [EmailAddress](#)
- [EmailAddressType](#)
- [EmergencyContact](#)
- [EmergencyContactRelationship](#)
- [Enterprise](#)
- [Genders](#)
- [Identity](#)
- [Language](#)
- [NationalId](#)
- [Person](#)
- [Phone Number](#)
- [PhoneNumberType](#)
- [Position](#)
- [SocialAccount](#)
- [SocialMedia](#)
- [State](#)
- [Title](#)
- [VisaType](#)
- [Worklist](#)

The properties for each entity are specific to it and only these properties should be used in [in subtyping](#) or [customization](#).

Address

The Address entity contains the following building blocks.

Lists

Name	Display Name	Description	Attributes
AllAddresses	All addresses	A list of all available addresses.	Address line 2 City Note Postal code Street address 1
AllAddressesInternal	All internal addresses	An internal list that is used by Identity package and that should not be replaced	ID ItemId City Note Postal code Street address 1
SelectDefault Address	Select default address	A list used to select a default address for an organizational unit.	ID City Postal code Street address 1
SelectWorkAddress	Select work address	A list used to select a work address for an organizational unit.	ID City Postal code Street address 1

Properties

Property	Display Name	Description	Data type	Length
City	City	The city for the address.	Text	64
Note	Note	Any notes about the address.	Long Text	
PostalCode	Postal code	The postal code for the address.	Text	16
StreetAddress	Address line 1	The first line of the street address.	Text	1024
StreetAddress2	Address line 2	The second line of the street address.	Text	128

Relationships

Name	Display Name	Description	Type	Target entity/ Child name	Pair with
County			To Peer	County	
ToCountry			To Peer	Country	ToAddress
To State			To Peer	State	ToAddress

Assignment

The Assignment entity contains the following building blocks.

Lists

Name	Display Name	Description	Attributes
AssignmentListInternal	Internal assignments	Internal list of assignments.	

Properties

Property	Display Name	Description	Data type	Length
End _Date	End date	The end date for the assignment.	Date and Time	
Principal	Principal	The principal assignee for the assignment.	Boolean (True or False)	
Start_Date	Start date	The start date for the assignment.	Date and Time	

Relationships

Name	Display Name	Type	Multiplicity	Target entity/ Child name	Pair with
toPersonToOne	Person to	To	To One	Person	ToAssignment

Name	Display Name	Type	Multiplicity	Target entity/ Child name	Pair with
	assignment	Peer			
toPositionToOne	Person to position	To Peer	To One	Position	ToAssignment

Rules

Name	Rule summary
DateCheck	Show warning "End date should be greater than start date." If item.Properties.End_Date < item.Properties.Start_Date

Country

The Country entity contains the following building blocks.

Action bar

Name	Buttons
CountryActionBar	History and Delete

Forms

Name	Display Name	Description
Create	Create	Used to create a country.
Default	Default	Used to display information about a country.

Layouts

Name	Display Name	Type	Forms used
Default_Layout	Default Layout	Full	Default
Preview	Preview	Preview	Default

Lists

Name	Display Name	Description	Attributes
AllCountries	All countries	List of available	Address line 2

Name	Display Name	Description	Attributes
		countries used to associate a country with an address.	City Note Postal code Address line 1
CountryListInternal	CountryListInternal	An internal list that is used by Identity package and that should not be replaced.	ID ItemId Country code Name

Properties

Name	Display Name	Description	Data type	Length
Country_Name	Country name	The name of the country.	Text	64
ISO_country_code	ISO country code	The ISO country code.	Text	2

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
State	State	To Child	To Many	State	ToCountry
ToAddress	To Address	To Peer	To One	Address	ToCountry

County

The County entity is a child of the State entity.

Lists

Name	Display Name	Description	Attributes
AllCounties	All counties	Default list that shows all the available counties	Name

Properties

Name	Display Name	Description	Data type	Length
Name	Name	Name of the county	Text	128

EmailAddress

The EmailAddress entity contains the following building blocks.

Lists

Name	Display Name	Description	Attributes
AllEmailAddresses	All email addresses	The list that can be used to browse email addresses. This list is not visible to the end user.	Address

Properties

Name	Display Name	Description	Data type	Length
Address	Address	Email address.	Text	128

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
Type	Type	To Peer	To One	EmailAddressType	-none-

EmailAddressType

EmailAddressType is a master data entity that can be created by using the Identity package.

Lists

Name	Display Name	Description	Attributes
AllEmailAddressTypes	All email addresses	List of available email address types.	Type

Properties

Name	Description	Data type	Length
Address	Email address.	Text	128

EmergencyContact

EmergencyContact is a child of the Person entity.

Lists

Name	Display Name	Description	Attributes
AllEmergencyContacts	All emergency contacts	Default list that shows all the emergency contacts.	Relationship

Properties

Name	Display Name	Description	Data type	Length
Dependent	Dependent	Yes or No to indicate whether the emergency contact is a dependent.	Boolean	
Details	Details	Details about the emergency contact.	Text	128

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
EmergencyContact	Emergency contact	To Child	To Many	EmergencyContact	- non e-
Relationship	Relationship	To Peer	To One	EmergencyContactRelationship	- non e-

EmergencyContactRelationship

The EmergencyContactRelationship entity serves as a pick list. Entries will be things such as Mother, Father, Grandmother, and so forth.

Forms

Name	Display Name	Description
Create	Create	Used to create a relationship for an emergency contact.
Default	Default	Used to display information about a relationship for an emergency contact.

Lists

Name	Display Name	Description	Attributes
AllEmergencyContactRelationships	All emergency contact relationships	List of all available emergency contact relationships.	Relationship

Properties

Name	Display Name	Description	Data type	Length
Relationship	Relationship	Relationship	Text	64

Enterprise

The Enterprise entity is an organizational unit type. It can be used a “root” for other organizational units.

Lists

Name	Display Name	Description	Attributes
AllEnterprises	All enterprises	List of all available enterprises.	

Genders

The Genders entity contains the following building blocks.

Action bar

Name	Buttons
GenderActionBar	History and Delete

Forms

Name	Display Name	Description
Create	Create	Used to create a gender.
Default	Default	Used to display information about a selected gender.
Preview_Form		

Layouts

Name	Display Name	Type	Forms used
Preview	Preview	Preview, Full	Preview Form

Lists

Name	Display Name	Description	Attributes
Genders_List	All genders	List of all available genders.	Gender
GendersListInternal	GendersListInternal	An internal list that is used by Identity package and that should not be replaced.	Id ItemId Gender

Properties

Name	Display Name	Description	Data type	Length
Gender	Gender	The name of the gender.	Text	16

Group

The Group entity contains the following building blocks.

Action bar

Name	Buttons
GroupsActionBar	Open

Forms

Name	Description
GroupsAndMembersForm	Used for Members tab in the Default layout. This form displays the list of members of the group that are synchronized from OTDS.
HeaderInformation	Used for Header information in the Default layout, which shows the information about the group.
MembersForm	Not used in the layout.
RolesForm	Not used in the layout.
SubGroupsForm	Used for the Subgroups tab in the Default layout.
SummaryForm	Used for the Summary tab in the Default layout.

Layouts

Name	Display Name	Type	Forms used
DefaultLayout	Default Layout	Full	HeaderInformation (for Header) Summary (for Summary tab) GroupAndMembersForm (for Members tab) Sub Groups Form (for Member Of tab)
Preview	Preview	Preview	Summary

Lists

Name	Display Name	Description	Attributes
AllGroupsListInternal	All internal groups	An internal list that is used by Identity package and that should not be replaced.	Description
GroupsList	All groups	List of available groups.	ID ItemId Description name member ID

Properties

Name	Display Name	Description	Data type	Length
Description	Description	A description of the group. This is a property of the Identity entity (base entity).	Text	256
Display_Name	Display Name	The display name of the group.	Text	64
memberid	Member ID	Used by the OTDS push connector.	Text	64
Name	Name	The name of the group. This is a property of the Identity entity (base entity).	Text	64

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
GroupToIdentity		To Peer	To Many	Identity	IdentityToGroup
IdentityToGroup		To Peer	To Many	Group	GroupToIdentity
IdentityToRole		To Peer	To Many	Role	RoleToIdentity
IdentityToWorklist		To Peer	To Many	Worklist	WorklistToIdentity

Identity

The Identity entity is the base entity of the Identity Package. Its properties are used by other entities in the package.

Lists

Name	Display Name	Description	Attributes
GroupList	All groups	List of the available groups.	
OrganizationalUnitList	All organizational	List of the available	

Name	Display Name	Description	Attributes
	units	organizational units.	
UsersList	All users	Default list of the available users.	

Properties

Name	Display Name	Description	Data type	Length
Description	Description		Text	256
Name	Name		Text	64

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
IdentityToGroup		To Peer	To Many	Group	GroupToIdentity
IdentityToRole		To Peer	To Many	Role	RoleToIdentity
IdentityToWorklist		To Peer	To Many	Worklist	WorklistToIdentity

Language

Language is a master data entity that has a relationship with the Person entity. It has a Create form that enables adding data for a title.

Forms

Name	Description
Create	Used to create a title for the language.
Default	Used to display information about a selected language.

Lists

Name	Display Name	Description	Attributes
AllLanguages	All languages	The list of available languages.	Name

Properties

Name	Display Name	Description	Data type	Length
IsoLanguageCode	ISO language code	Language code.	Text	15
Name	Name	Name of the language (such as English, Spanish, and so forth).	Text	128

NationalId

NationalId is a child of the Person entity.

Properties

Name	Display Name	Description	Data type	Length
CardType	Card Type	The type of card.	Text	256
DocumentNumber	Document number	The document number for the ID card.	Text	256
IsPrimary	Is primary	Whether the ID is the primary ID.	Boolean	
DateOfIssue	Date of issue	The date when the ID card was issued.	Date	
DateOfExpiration	Date of expiration	The expiration date for the ID card.	Date	

Lists

Name	Display Name	Description	Attributes
AllNationalIDs	All national IDs	List of available national IDs	<ul style="list-style-type: none"> • Card type • Date of expiration • Date of issue • Document number • Is primary

Organizational unit

The Organizational unit entity describes an organizational unit within the enterprise. For example, an enterprise called Acme Distributors may contain organizational units such as Human Resources, R&D, Sales, and so forth

Action bar

Name	Buttons
UnitActionBar	History and Delete

Forms

Name	Description
Create	Used to create an organizational unit.
Default	Used to display information about a selected organizational unit.
HeaderInformation	Used to display header information about an organizational unit.
Members_Form	Used to maintain members of the organizational unit.
Positions	Used to maintain positions of the organizational unit.
SubUnitsForm	Used to maintain subunits of the organizational unit.
Summary	Used to maintain summary information about the organizational unit.

Layouts

Name	Display Name	Type	Forms used
Default_Layout	Default Layout	Full	HeaderInformation (for Header) Summary (for Summary tab) Positions Form (for Positions tab) Members Form (for Members tab) SubUnitsForm (for Subunits tab)
Preview	Preview	Preview	Summary

Lists

Name	Display Name	Description	Attributes
OrganizationListInternal	OrganizationListInternal	An internal list of organizational units.	

Name	Display Name	Description	Attributes
		Not visible to users. Should not be replaced.	
OrgUnitList	All organizational units	A list of available organizational units.	Type
selectUnit	Select unit(s)	Used in the Worklist entity for browsing Units to associate.	Name

Properties

Name	Display Name	Description	Data type	Length
Description	Description	A description of the organizational unit. This is a property of the Identity entity.	Text	256
Name	Name	The name of the organizational unit.	Text	64
OrgUnit_FQN	Organizational unit name	The full qualified name for the organizational unit. This is a property of the Identity entity.	Long Text	
OrgUnit_RuntimeId	Organizational unit runtime ID	The Runtime ID for the organizational unit.	Long Text	
Type		The type of the organizational unit: Hierarchical	Enumerated Text	64

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
Addresses		To Peer	To Many	Address	-none-
DefaultAddress		To Peer	To One	Address	-none-

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
EmailAddresses		To Peer	To Many	EmailAddress	-none-
IdentityToGroup		To Peer	To Many	Group	GroupToIdentity
IdentityToRole		To Peer	To Many	Role	RoleToIdentity
IdentityToWorklist		To Peer	To Many	WorkList	WorklistToIdentity
PhoneNumbers		To Peer	To Many	PhoneNumber	-none-
Position		To Child	To Many	Position	ToOrganizationalUnit
PrimaryEmailAddress		To Peer	To One	EmailAddress	-none-
PrimaryPhoneNumber		To Peer	To One	PhoneNumber	-none-
SocialAccounts		To Peer	To Many	SocialAccount	-none-
SubUnits		To Peer	To Many	OrganizationalUnit	-none-

Rules

Name	Rule summary
OnCreate	Start Process BPMs/CreateTeamEntryInNotificationRepository When a new item is created
onDelete	Start Process BPMs/DeleteTeamEntryInNotificationRepository When an item is deleted
UnitNameMandatory	Show error "Organization unit name is mandatory" If item.Properties.Name is null When a new item is created

Person

For every user of the system, both a User entity and a Person entity are created. Users are people that are users of Process Suite. The User entity is a small entity, as all personal information is stored in the Person entity. For people who are part of applications (and their data), the Person information is not synchronized from OTDS through the OTDS push connector. For example, your applications may need to register people for different purposes such as the following:

- Contact person of an enterprise
- Claimant of a claim
- Employee

Action bar

Name	Buttons
PersonActionBar	Open, Delete, and History

Forms

Name	Description
Create	Used to create a person.
HeaderInformation	Used to display header information about a person.
Summary	Used to maintain summary information about a person.

Layouts

Name	Display Name	Type	Forms used
DefaultLayout	DefaultLayout	Full	HeaderInformation SummaryForm
UserLayout	Preview	Preview	Create

Lists

Name	Display Name	Description	Attributes
AllPersons		A list of available persons.	
PersonList	All contacts	A list from which a user can select a person.	Display Name Email Is internal

Name	Display Name	Description	Attributes
PersonListInternal	PersonListInternal	An internal list that is used by the Identity package and that should not be replaced during customization.	
SelectUser	Select user	An internal list that is used by the Identity package and that should not be replaced during customization.	Display Name Email Is internal User ID

Properties

Name	Display Name	Description	Data type	Length
Birthdate	Date of birth	The person's date of birth in yyyy-mm-dd format.	Date	
BirthName	Birth name	The person's birth name.	Text	128
Department	Department	Not used	Text	64
DisplayName	Display Name	The display name for the person.	Text	64
Email	Email	The person's email address.	Text	64
facsimileTelephoneNumber	Fax	The person's fax number	Text	64
FirstName	First name	The person's first name.	Text	64
Initials	Initials	The person's initials	Text	16
Is_Internal			Boolean	
LastName	Last name	The person's last name.	Text	64
MaritalStatus	Martial status	The person's marital status	Enumerated Text	

Name	Display Name	Description	Data type	Length
		(Single, Married, Widow, Divorced).		
MaritalStatusSince	Marital status since	The date when the person's marital status became effective.	Date	
memberid	Member ID	The person's member ID.	Text	64
MiddleName	Middle name	The person's middle name.	Text	64
Mobile	Mobile	The person's mobile number.	Text	16
notes	Notes	Additional information about the person.	Text	64
oTCompany	Company	Mapped to the Company attribute of OTDS.	Text	64
oTDepartment	Department	Mapped to the Department attribute of OTDS.	Text	64
PartnerName	Partner name	The name of the person's partner.	Text	128
PartnerNamePrefix	Partner name prefix	The prefix of the person's partner name.	Text	16
Phone	Phone	The person's phone number.	Text	16
physicalDeliveryOfficeName	Office	The name of the office where deliveries can	Text	64

Name	Display Name	Description	Data type	Length
		be made for the person.		
Picture	Picture	An image of the person that can be uploaded at runtime. If no image is provided, a default image will be displayed.	Image	
PreferredName	Preferred name	The name by which the person prefers to be addressed.	Text	128
Prefix	Prefix	The prefix for the person's name (none, Mr, Ms, Mrs).	Enumerated Text	
SecondLastName	Second last name	The person's second last name.	Text	128
Suffix	Suffix	The suffix for the person's name.	Text	16
title	Job title	The person's title.	Text	64
User_ID	User ID	The person's user ID.	Text	64

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
Addresses		To Peer	To Many	Address	-none-
CountryOfBirth		To	To One	Country	-none-

Name	Display Name	Type	Multiplicity	Target entity/ Child name	Pair with
		Peer			
EmailAddresses		To Peer	To Many	EmailAddress	-none-
EmergencyContact		To Child	To Many	EmergencyContact	-none-
NationalId		To Child	To Many	NationalId	-none-
PersonToGender		To Peer	To One	Genders	-none-
PersonToUser		To Peer	To One	User	toPerson
PhoneNumbers		To Peer	To Many	PhoneNumber	-none-
PreferredLanguage		To Peer	To One	Language	-none-
PrimaryEmailAddress		To Peer	To One	EmailAddress	-none-
PrimaryEmergencyContact		To Peer	To One	EmergencyContact	-none-
PrimaryPhoneNumber		To Peer	To One	PhoneNumber	-none-
RegionOfBirth		To Peer	To One	State	-none-
SecondTitle		To Peer	To One	Title	-none
SocialAccounts		To Peer	To Many	SocialAccount	-none-
ToAssignment		To Peer	To Many	Assignment	toPersonToOne
VisaType		To Peer	To One	VisaType	-none-
WorkAddress		To Peer	To One	Address	-none-

Rules

Name	Rule summary
OnDelete	Start Process BPMs/DeletePersonAddresses If item.Properties.Is_Internal==true When an item is deleted
RuleOnBirthdate	Show error "Birth date should not be future date." If item.Properties.Birthdate>=today

Phone Number

The PhoneNumber entity contains the following building blocks.

Lists

Name	Display Name	Description	Attributes
AllPhoneNumbers		A list of available phone numbers.	

Properties

Name	Display Name	Description	Data type	Length
Extension	Extension	Extension of the phone number	Text	64
Number	Number	Phone number	Text	64

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
Type		To Peer	To One	PhoneNumberType	-none-

PhoneNumberType

The PhoneNumberType entity serves as a pick list. Entries will be things such as Home, Work, Mobile, and so forth. which can be added by Identity package.

Forms

Name	Description
Create	Used to create a phone number type.
Default	Used to display details about a selected phone number.

Lists

Name	Display Name	Description	Attributes
AllPhoneNumberTypes		A list of available phone number types.	Type

Properties

Name	Display Name	Description	Data type	Length
Type	Type	Type of the phone number.	Text	128

Position

Position is a child entity of the Organizational Unit entity.

Forms

Name	Description
Create	Used to create a position.
Default	Used to display information about a selected position.

Lists

Name	Display Name	Description	Attributes
PositionList		A list of positions.	
PositionListInternal		An list of internal positions	

Properties

Name	Display Name	Description	Data type	Length
Description		A description of the position.	Text	64
Lead		Whether this is a lead position	Boolean (True or False)	
PositionName		The name of the position.	Text	64

Relationships

Name	Display Name	Type	Multiplicity	Target entity / Child name	Pair with
ToAssignment		To Peer	To Many	Assignment	toPositionToOne
ToOrganizationalUnit		To Parent	To One	Organizational Unit	Position
ToWorklist		To Peer	To One	WorkList	ToPosition

Role

The Role identity contains the following building blocks.

Action bar

Name	Buttons
RolesActionBar	Open

Forms

Name	Description
GroupsForm	
HeaderInformation	Used to display header information about a role.
MembersForm	Used to display the members of a role.
SummaryForm	Used to maintain summary information about a role.
UsersForm	Used to display the users assigned to the role.

Layouts

Name	Display Name	Type	Forms used
Default_Layout	Default Layout	Full	HeaderInformation (for Header) SummaryForm (for Summary tab) MembersForm (for Members tab)
Role_Preview_Layout		Preview	Preview

Lists

Name	Display Name	Description	Attributes
AllRolesInternal		An internal list that is used by the Identity package and that should not be replaced.	ID ItemID Description Member ID Name
RoleList	All roles	A list of available roles.	Description

Properties

Name	Display Name	Description	Data type	Length
Description		A description of the role. This is a property of the Identity entity (base entity).	Text	256
memberid	Member ID	Used by the OTDS push connector.	Text	64
Name		The name of the role. This is a property of the Identity entity (base entity).	Text	64
Package_Name	Package Name		Text	64

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
IdentityToGroup		To Peer	To Many	Group	GroupToIdentity
IdentityToRole		To Peer	To Many	Role	RoleToIdentity
IdentityToWorklist		To Peer	To Many	Worklist	WorklistToIdentity
RoleToIdentity		To Peer	To Many	Identify	IdentityToRole

SocialAccount

The SocialAccount entity provides the following building blocks.

Lists

Name	Display Name	Description	Attributes
AllSocialAccounts		A list of all social accounts	

Properties

Name	Display Name	Description	Data type	Length
InstantMessagingId	Instant Messaging ID	ID on the selected social media.	Text	128
URL	URL		Text	256

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
Domain		To Peer	To One	SocialMedia	-none-

SocialMedia

The SocialMedia entity serves as a pick list. Entries will be things such as LinkedIn, Facebook, Skype, and so forth.

Forms

Name	Description
Create	Used to create social media information.
Default	Used to display information about a selected social media.

Lists

Name	Display Name	Description	Attributes
AllSocialMedias	All social medias	List of all social media.	<ul style="list-style-type: none"> • Name

Properties

Name	Display Name	Description	Data type	Length
Name	Name	The name of the social media.	Text	128

State

State is a child of the country entity.

Lists

Name	Display Name	Description	Attributes
AllStates	All states.	List of states	<ul style="list-style-type: none"> • Name
AllStatesInternal		Internal list of states	

Properties

Property	Description
Name	The name of the state.

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
County	County	To Child	To Many	County	-none-
ToAddress	Address	To Peer	To Many	Address	ToState
ToCountry	ToCountry	To Parent	To One	Country	State

Title

Title is a master data entity that has a relationship with the Person entity. The entity has a Create form that enables adding data for a title.

Forms

Name	Description
Create	Used to create a title.
Default	Used to display information about a selected title.

Lists

Name	Display Name	Description	Attributes
AllTitles	All titles	Lists the available titles.	Name

Properties

Name	Display Name	Description	Data type	Length
Name	Name	Name of the title (such as B.S., M.D., Ph. D, and so forth).	Text	128

User

User is an internal system user who has logon credentials and who also has access to the application to track information such as preferences and history information.

Action bar

Name	Buttons
UsersActionBar	Open

Forms

Name	Description
GroupsForm	Used to add a user to a group.
HeaderInformation	Used to display header information about a user.
Preview	Used to view information about a user.
RolesForm	Used to add a user to a role.
SummaryForm	Used to maintain summary information about a user.

Layouts

Name	Display Name	Type	Forms used
DefaultLayout		Full	HeaderInformation((For Header) SummaryForm (for Summary tab) RolesForm (for Roles tab) GroupsForm (for Groups tab)
User_Preview_Layout	Preview	Preview	Preview

Lists

Name	Display Name	Description	Attributes
AllUsers	All users	A list of available users.	Full Name User Id
AllUsersInternal		An internal list that is used by Identity package and that should not be replaced.	Display Name Email Is Internal

Properties

Name	Display Name	Description	Data type	Length
Description		A description of the user. This is a property of the Identity entity (base entity).	Text	256
FullName	Full name	Full name of the user that is mapped to the OTDS attribute for User Name.	Text	128
memberid	Member ID	The user's member ID. This is for internal use only by the Identity component.	Text	64
Name		The user's name. This is a property of the Identity entity (base entity).	Text	64
PreferredLocale	Preferred locale	Preferred locale of the user.		
UserId	User ID	A unique ID for the user.	Text	64

Relationships

Name	Display Name	Type	Multiplicity	Target entity/ Child name	Pair with
IdentityToGroup		To Peer	To Many	Group	GroupToIdentity
IdentityToRole		To Peer	To Many	Role	RoleToIdentity
IdentityToWorklist		To Peer	To Many	WorkList	WorklistToIdentity
Personalization		To Child	To Many	Personalization	-none-
toPerson		To Peer	To One	Person	PersonToUser

VisaType

The VisaType entity serves as a pick list. Entries will be things such as KAZ-Identity Card, Domestic Passport, Passport, POL-ID Card, International Passport, POL-Passport, Taxpayer Identification Number (INN), TWN-Work Permit for Foreign Professionals, and so forth.

Forms

Name	Description
Create	Used to add a visa type.
Default	Used to display information about a selected visa type.

Lists

Name	Display Name	Description	Attributes
AllVisaTypes	All visa types	A list of available visa types.	Name

Properties

Name	Display Name	Description	Data type	Length
Name	Name	Name	Text	128

Worklist

If you define worklists in BPM processes, application users with the appropriate access can link a worklist to a team. For example, if you have a worklist for complex cases and another worklist for simple cases, each can be linked to a different team with the appropriate skill level to resolve the case.

Action bar

Name	Buttons
WorklistActionBar	History and Delete

Forms

Name	Description
Create	Used to create a worklist.
HeaderInformation	Used to display header information about a worklist.
SummaryForm	Used to maintain summary information about a worklist.
UnitForm	Used to associate available Units to worklist and is used for Units tab in the Default layout.
WorklistManagerForm	Used to associate a position as a worklist manager and for the Lead Position(s) tab in Dafult layout.

Layouts

Name	Display Name	Type	Forms used
DefaultLayout	Default Layout	Full	HeaderInformation (for Header) SummaryForm (for Summary tab) UnitForm (for Units tab) WorklistManagerForm (for Lead position(s) tab)
Preview	Preview	Preview	SummaryForm

Lists

Name	Display Name	Description	Attributes
AllWorklistInternal		An internal list that is used b the Identity package and that should not be replaced.	Description
Worklists	All worlists	A list of available worklists.	Id ItemId Description Worklist name

Properties

Name	Display Name	Description	Data type	Length
Description		A description of the worklist.	Text	64
WorkListName		The name of the worklist.	Text	128

Relationships

Name	Display Name	Type	Multiplicity	Target entity/Child name	Pair with
ToPosition		To Peer	To Many	Position	ToWorklist
WorklistToIdentity		To Peer	To Many	Identity	IdentityToWorklist

Rules

Name	Rule summary
OnCreate	Start Process BPMs/CreateWorklistEntryInNotificationRepository When a new item is created

Identity roles

The Identity package provides the following roles:

- Identity Administrator
- Identity Push Service
- Identity User

OTDS

The following sections describe how OTDS is used with the Identity Package.

Information pushed from OTDS to Process Platform

The following entities in the Identity Package are synchronized from OTDS:

- User
- Role

- Group

User, Role, and Group information is directly synchronized with OTDS using the OTDS Push Connector for OpenText CARS and Identity components. In addition, different relationships between User, Role, and Group are also saved to the User, Role, and Group entities that are created.

If a new user is added in OTDS, the user becomes available in CARS, Entity Modeling, and the Process Experience user interface. Synchronization from OTDS to Process Platform is one-way. The OTDS Push connector will only synchronize information from OTDS to Process Platform. That is, you cannot add users in Process Experience and synchronize them with OTDS.

See the *Process Platform Process Experience User's Guide* for details about how to administer the identities.

Managing information pushed to OTDS

Data is only pushed automatically from OTDS to Process Platform and not from Process Platform to OTDS. There are situations when data must be synchronized from Process Platform to OTDS. In these situations the OTDS Configurator tool should be run.

These situations are when a functional or application package role is created in CARS. That is the case whenever:

- A package with a functional or application role is deployed in Process Platform.
- An updated package is deployed and the package has new, renamed, or updated roles. A role is updated when either the role – role membership is changed.
- Process Platform is upgraded to a newer version / fixpack as there may be new roles or updated roles.
- A package is undeployed from Process Platform. To have OTDS configurator delete the roles from OTDS a specific property needs to be set when calling OTDS configurator (`otdsconfigurator.delete.roles=true`, default is `false`). see OTDS Configurator for Process Platform.
- A CWS project containing roles is published.

Note: The `otdsconfigurator` utility cannot detect a role name change. Therefore, when a role is renamed, the user and group membership of that role is lost. You must map the role to the user and group membership again.

If you need to manually synchronize package roles to OTDS, you can use the OTDS Configurator command line utility (`otsconfigurator`). For information, see the topics called [OTDS Push Connector for OpenText CARS and Identity components](#) and [Using OTDS for Process Platform user management](#) in the Process Platform online help.

Mapping of OTDS identities to Identity entities

The following tables show how OTDS Identities are mapped to Identity Package entities.

User

OTDS Attribute	Entity to which it is mapped	Entity Modeling property name
birthDate Note: Should be entered as (YYYY-MM-DD) in OTDS	Person	Birthdate
City	Address	City
Company	Person	oTCompany
Country/Region	Country	Name
Department	Person	oTDepartment
Department	Person	Department
Description	Identity	Description
Display Name	Person	Display Name
Email	Person	Email
Fax	Person	facsimileTelephoneNumber
First Name	Person	FirstName
gender	Gender	Gender
Initials	Person	MiddleName
Job Title	Person	title
Last Name	Person	LastName
Notes	Person	notes
Office	Person	physicalDeliveryOfficeName
oTExternalID3	User	User ID
oTExternalID3	Person	User ID
oTExternalID3	Identity	Name
oTMobile	Person	Mobile
Phone	Person	Phone
State/Province	State	State/Province
Street	Address	Street Address
User Name	User	FullName
Zip/Postal Code	Address	Postal Code

Group

OTDS Attribute	Entity to which it is mapped	Entity Modeling property name
Description	Identity	Description
Display Name	Group or Role	Display Name
Group Name	Identity	Name

Customizing the Identity package

To make the OpenText Entity Identity Components visible and customizable, right-click **Imported Application Packages > Show Platform Packages**. To customize the Identity package, see [Customizing an application](#). When you save the customized entity, the default destination is the first project in the workspace.

Chapter 11

Using the expression editor

The expression editor is a tool that enables you to create expressions that are used for making calculations based on the data in your application. It can be used in Basic or Advanced mode. Basic mode is designed to be easy to use, while Advanced mode is more powerful and enables expressing more complex conditions. You can design filters in Basic mode and, if necessary, switch to Advanced mode to edit expressions in text mode. The expression editor is used by many building blocks, such as [Rules](#), [Deadlines](#), [Lifecycle](#), and [Activity flow](#).

Basic mode provides the following options:

- In the **If** list, select **all** or **one** to specify whether to show results if all or one of the terms are true.
- In the **Select** list, define the terms by selecting a property, operator, and value.
- Use Plus  (Plus) and Minus  (Minus) to add and remove conditions.

Dates in expressions

There are two ways to work with dates in expressions. The first is to use the date keyword and pass in the individual parts of the date. For example the date(2015, 12, 25) keyword always returns December 25th 2015. There are few use cases for hard coding a specific date in an expression, but it can be necessary at times. The most common use case is to use the "now" and "today" keywords. The benefit of using these keywords is that the dates are not hard-coded in the expression. These keywords use the current date and time in your expressions at the time of evaluation as opposed to a hard-coded date. The today keyword returns the current date and the now keyword returns the current date and time.

Enumerated values in expressions

If the expression involves an enumerated value, it should include the internal value instead of the display name. The display name may change depending on the locale so the internal value is required.

Expression language

Expressions enable you to specify logical, arithmetic, and text processing operations. Expressions are most frequently used to construct business rules where a conditional expression is combined with an action. This section provides a number of topics that explain various aspects of expressions.

A rule or expression is a series of operators and operands that, when evaluated, results in a value. Following are some sample expressions that illustrate the range of supported functionality.

Expression	Notes
3 * (2 + 7)	Simple integer arithmetic with parenthesis
75.3 * (ivar + 7)	Decimal numbers and variables, with type coercion for mixed number types
ivar == 3 "word" != svar	Boolean comparison and logical operators such as AND and OR. In many cases, there are several synonymous operators. For example, this expression could also be written: ivar == 3 OR "word" != svar
substring("string", 2, 6) + "test"	A variety of functions and methods are supported using the standard function (argument, argument, ...) syntax
length(svar+'test') * 7	Function arguments may be expressions
'the brown fox' contains 'br[l-p]wn'	Complex string operations are supported, in this case regular expression pattern matching
true && 'true'	Automatic type coercions (this example shows a string coerced into a boolean value)
now + 1 end of months + 2 mondays	Complex date and time calculations (this example computes the 2nd monday of next month).
0b01010101 & 0xf0 0o10	Binary, hexadecimal, and octal integer constants, and bitwise operators are all supported
'abc' in list{'abc', 'ghi', 'jkl'}	Array data types are supported
'abc' in list{'abc', 'ghi', svar +'jkl'}	Array constant entries may be expressions
'def' in range{'abc', 'ghi'}	Range data types are supported
bvar ? ivar * 17 : ivar * 7	The trinary condition ? true-value : false-value

Expression	Notes
	conditional "if" operator is supported
(ivar>=37 ? ivar+77 : (ivar+77)%37) * 17 < 44	The result of a trinary "if" operator may be used as an expression term (note the parenthesis)
svar in (bvar ? list{"abc", "g"+"hi"} : list{"xyz", "uvw", "rst"})	An "if" that determines if a value is in a list of possible values

Properties

The information manipulated by expressions is accessed by referencing properties of items that are available in the context in which the expression is being evaluated. The context - the set of available items - varies depending on where the expression is being used. The items that are usually in context are listed below. Each entity has its own set of properties and methods, accessed using the property access operator '.'.

- item - The item that defines the current context, this is often called the current item.
- user - The current user; an instance of the system defined User entity.

Note: Properties of data type Image cannot be used in expressions.

Every property has a fully qualified name that includes the names of the item and the building block where the property is defined. The following examples are in the context of an invoice item with the structure shown (this is a highly simplified entity with a small set of properties).

```
invoice

    properties

        accountNumber

        customerName

        amount

line

    properties

        partNumber

        description

        quantity

        price

        total

        taxable
```

A show error rule in the line entity could use the following condition to ensure that the total value for each line item is correct.

```
Show error "Incorrect total" if item.properties.price * item.properties.quantity != item.properties.total
```

This can be expressed a bit more succinctly using shorthand property names as follows:

```
Show error "Incorrect total" if price * quantity != total
```

Expressions can use aggregation operations on "to many" relationships (see [Arrays and aggregation](#)). The following example shows Show error rule on the invoice entity to validate that the total amount on the invoice is equal to the sum of all the line item totals.

```
Show error "Incorrect total" if sum(line[].total) != amount
```

Decimal values

The Decimal property type explicitly specifies the scale (the number of digits maintained to the right of the decimal). For most operations (addition, subtraction) this is simple, as the resulting number ends up with the maximum scale of the two operands.

But for division, some "arbitrary" decisions need to be made about how to handle scale, and whatever arbitrary decision is made, it is not going to be proper for some calculation, done in some circumstance, that requires greater scale. The scale of the result of a Decimal

division is the scale of the numerator plus four. The following examples illustrate the behavior of decimal division in various cases.

Expression	Result
value = 2034183616.0 / 100000	20341.83616 (as expected)
value = 2034183616 / 100000	20341 (this is actually integer division followed with a cast to Decimal)
value = 2034183616.0 / 100000.0	20341.83616 (as expected)
value = 2034183616 / 100000.0	20341.8362 (surprise!)

The fourth case was rounded to four decimal places because the numerator's scale is 0 (it is an integer) to which 4 is added, with the result's scale therefore set to 4 digits. The actual result has 5 digits of scale and is rounded to fit.

If you need more precision, you can control this by forcing the desired scale on the numerator. For example, to force the result of the division to have at least a precision of 7 digits, do the following:

```
value = (numerator + 0.000) / denominator
```

The addition operator forces the numerator to have at least a scale of 3, and then the division operator adds 4 to the scale in its result, to give a final scale of 7.

Special characters in string constants

String constants are delimited using either the single quote ('') or double quote ("") symbol. The backslash character (\) is used inside string constants as an escape character. The following table lists the supported escape character sequences in string constants.

Escape Sequence	Resulting Character
\\	\
\n	newline (0x0a)
\r	carriage return (0x0d)
\t	tab (0x09)
'	single quote
"	double quote

Null values

Null values are handled and propagated through expression evaluation. Variable value resolution may return null values, which may mean that the final expression value is null.

This will produce various effects, depending on where the expression is used. The following sample rule illustrates this behavior.

```
SET Properties.X = 1 IF Properties.Status != 4
```

You can expect this condition to evaluate to true if status is null (after all null is not equal to 4). However this is not what will happen. The expression engine is designed to propagate the fact that status is "unknown" back through all computations. So the expression status != 4 evaluates to null when status is null. The boolean expression returns null which the rule manager treats as false, and the action of the rule is not executed.

When writing expressions you need to be aware of, and account for, null values. This example can be "corrected" by using the optional operator as shown below. The optional operator specifies that the value of the first parameter is optional (may be null), and if it is not available, to return the default value specified by the second parameter.

```
SET Properties.X = 1 IF Optional(Properties.Status != 4, true)
```

Built-in functions

The expression language includes a number of useful built-in functions that are described in the following sections. For details, see [Arrays and aggregation](#).

average and averageN

This function returns the average of all values in the list. It accepts lists of integer, long, decimal, or date values. Any null values are ignored.

```
average = average(list)
```

The null aware version averageN returns null if any value in the list is null.

countTrue and countTrueN

This function returns the number of true values in a specified list. This function accepts only Boolean lists.

```
count = countTrue(list)
```

The null aware version countTrueN returns a range of possible true counts, with the lower value the number of entries in the list that are currently true, and the higher value the currently true count, plus the count of entries that are currently null.

max and maxN

This function returns the maximum value in the list. It accepts lists of integer, long, decimal, or date values. Any null values are ignored.

```
max = max(list)
```

The null aware version maxN returns null if any value in the list is null.

median and medianN

This function returns the median of all values in the list. It accepts lists of integer, long, decimal, or date values. Any null values are ignored.

```
median = median(list)
```

The null aware version medianN returns null if any value in the list is null.

min and minN

This function returns the minimum value in the list. It accepts lists of integer, long, decimal, or date values. Any null values are ignored.

```
min = min(list)
```

The null aware version minN returns null if any value in the list is null.

mode and modeN

This function returns the mode of the specified list. The mode is the most frequently occurring value in the list. This function supports integer, long, decimal, or date lists. If two values occur the same number of times, it returns the value that occurs first in a sorted list of the values. If no value occurs more than once, it returns null. Any null values are ignored.

```
mode = mode(list)
```

The null aware version modeN may return a value even if nulls exist in the list, if the leading candidate has an overwhelming lead (if it occurs more frequently than the next leading candidate plus the number of null values).

optional

This function can be used to indicate that a value or expression result is optional - that a null value is valid. The following example shows how this can be used. Without the optional function, a null Middle value would result in the entire expression having a null value.

```
Set Name = First + optional(Middle) + Last
```

The following table lists the available overloads.

Function	Arg	Type	Returns
optional(arg)	arg	any	If arg is not null, returns arg, otherwise depends on the variable's type: string - empty string integer, decimal, or long - 0 boolean - false date/time - Midnight (12:00am) January 1, 1970
optional(arg, default)	arg any	default any	If arg is null, returns the specified default value.

random

This function returns a random integer within the specified range.

Function	Arg	Type	Returns
random(low, hi)	low hi	integer integer	Returns a random number between low and hi (inclusive)

sum and sumN

This function returns the sum of all values in the list. This function accepts lists of integer, long, or decimal values. Any null values are ignored.

```
sum = sum(list)
```

The null aware version sumN returns null if any value in the list is null.

isInGroup

This function returns true if the current user is a member of the organizational role/group that is specified by the string groupName.

```
isInGroup("groupName")
```

isInRole

This function returns true if the current user is a member of the package role that is specified by the strings package and roleName.

```
isInRole("package", "roleName")
```

Operator reference

The following table lists all of the operators supported by the pre-loaded grammars. The table shows operators in precedence order (with a gray header row separating each level of precedence) and for operators of equal precedence, shows order of associativity.

Note: A programmer can extend the expression language as needed.

Operator	Description	Usage example	Data type(s)
Keywords and functions			
blank	empty, zero-length string	blank	string
true	boolean constant	true	boolean
false	boolean constant	false	boolean
now	current date and time	now	date
today	current date	today	date
never	a null date	never	date
range	create a range (inclusive)	range{arg1, arg2}	integer, long, decimal, string, or date
date	create a date	date(arg1, ..., argn)	All arguments are integers. Returns a datetime (if the time is not specified it is set to midnight; 12:00am). Can be in the form: <ul style="list-style-type: none">• date(year, month, day)• date(year, month, day, hour, minute)• date(year, month, day, hour, minute, second)

Operator	Description	Usage example	Data type(s)
Keywords and functions			
			minute, second)
time	create a time	time(arg1 ...)	All arguments are integers. Returns a datetime with the date set to January 1, 1970). Can be in the form: <ul style="list-style-type: none"> • time(hour, minute) • time(hour, minute, second)
function()	function call		See Built-in functions for a list of available functions.
Unary and coercion operators (<unary-op>) - evaluated right to left			
!	logical not	!arg	boolean
not	logical not	not arg	boolean
-	unary minus	-arg	integer, long, or decimal
~	bitwise complement	~arg	integer or long
(string)	coerce to a string	(string) arg	coerce boolean, integer, long, decimal, or date to a string (returns null only if the argument is null)
(decimal)	coerce to a decimal	(decimal) arg	coerce boolean, integer, long, string, or date to a decimal (returns null if the coercion fails)
(long)	coerce to a long	(long) arg	coerce boolean, integer, decimal, string, or date to a long (returns null if the coercion fails)
(integer)	coerce to an integer	(integer) arg	coerce boolean, long, decimal, string, or date to an integer (returns null if the coercion fails)
(boolean)	coerce to a boolean	(boolean) arg	coerce integer, long, decimal, string, or date to a boolean (returns null if the coercion fails)
(date)	coerce to a date	(date) arg	coerce string to a date (returns null if the coercion fails)

Operator	Description	Usage example	Data type(s)
Keywords and functions			
Arithmetic operators (<binary-op>) - processed left to right			
*	multiply	arg1 * arg2	integer, long, or decimal
/	divide	arg1 / arg2	integer, long, or decimal
%	modulo	arg1 % arg2	integer, long, or decimal
+	add	arg1 + arg2	integer, long, decimal, or string (concatenation)
-	subtract	arg1 - arg2	integer, long, or decimal
+ time-unit	add time-unit	arg1 + arg2 minutes	First arg is date, second arg is integer, returns date, see list of time-units
- time-unit	subtract time-unit	arg1 - arg2 mondays	First arg is date, second arg is integer, returns date, see list of time-units
at	set the time of a date/time	arg1 at arg2	First arg is date, second arg is date, returns date with date of arg1 and time of arg2
Relational operators			
<	less than	arg1 < arg2	integer, long, decimal, string (case sensitive), or date
>	greater than	arg1 > arg2	integer, long, decimal, string (case sensitive), or date
<=	less than or equal to	arg1 <= arg2	integer, long, decimal, string (case sensitive), or date
>=	greater than or equal to	arg1 >= arg2	integer, long, decimal, string (case sensitive), or date
[is] in	list or range containment	arg1 in arg2	integer, long, decimal, string and date, second argument is a list or range
[is] not in	list or range containment	arg1 not in arg2	integer, long, decimal, string and date, second argument is a list or range
contain[s] string	containment with pattern matching	arg1 contains arg2	string arguments, returns boolean

Operator	Description	Usage example	Data type(s)
Keywords and functions			
does not contain string	containment with pattern matching	arg1 does not contain arg2	string arguments, returns boolean
==	equality	arg1 == arg2	integer, long, decimal, string (case sensitive), or date
=	equality	arg1 = arg2	integer, long, decimal, string (case sensitive), or date (when used in a script statement, this is recognized as the assignment operator, not an equality comparison)
is	equality	arg1 is arg2	integer, long, decimal, string (case sensitive), or date
== null	equality null test	arg1 == null	integer, long, decimal, string, or date
= null	equality null test	arg1 = null	integer, long, decimal, string, or date
is null	equality null test	arg1 is null	integer, long, decimal, string, or date
!=	inequality	arg1 != arg2	integer, long, decimal, string (case sensitive), or date
<>	inequality	arg1 <> arg2	integer, long, decimal, string (case sensitive), or date
is not	inequality	arg1 is not arg2	integer, long, decimal, string (case sensitive), or date
!= null	not null test	arg1 != null	integer, long, decimal, string, or date
<> null	not null test	arg1 <> null	null integer, long, decimal, string, or date
is not null	not null test	arg1 is not null	integer, long, decimal, string, or date
Bitwise operators (<binary-op>) - processed left to right			
&	bitwise and	arg1 & arg2	integer or long
^	bitwise exclusive or	arg1 ^ arg2	integer or long

Operator	Description	Usage example	Data type(s)
Keywords and functions			
	bitwise inclusive or	arg1 arg2	integer or long
Logical operators (<binary-op>) - processed left to right			
&&	logical and	arg1 && arg2	boolean (arg2 is not evaluated if arg1 is false)
and	logical and	arg1 and arg2	boolean (arg2 is not evaluated if arg1 is false)
	logical or	arg1 arg2	boolean (arg2 is not evaluated if arg1 is true)
or	logical or	arg1 or arg2	boolean (arg2 is not evaluated if arg1 is true)
Trinary operators (<trinary-op>) - processed from right to left			
? :	conditional "if"	arg1 ? arg2 : arg3	arg1 is boolean, arg2 and arg3 may be any type (including array and range). If arg1 is true, the value is arg2, otherwise arg3.

Date methods

A number of built-in date methods are provided to simplify working with dates. New dates can be created using the date and time functions. The at operator can be used to update the time portion of a date. These are illustrated in the following table (the parameters used in these methods are described following the table).

Expression	Comment
date(year, month, day)	A date with the specified year, month, and day.
date(year, month, day, hour, minute)	A date with the specified year, month, and day and time.
date(year, month, day, hour, minute, second)	A date with the specified year, month, and day and time including seconds.
dateEaster(year)	The date of the Easter holiday for the given year. This method is provided because the calculation needed is complex enough to warrant a built-in method.
time(hour, minute)	A time with the specified hour and minute (hours

Expression	Comment
	are specified as a 24 hour clock).
time(hour, minute, second)	A time with the specified hour, minute and second.
now at time(20, 00)	Today's date at 8:00pm.
date(timezone, locale, year, month, day)	A date with the specified year, month, and day, in the specified timezone and locale.
date(timezone, locale, year, month, day, hour, minute)	A date with the specified year, month, and day and time, in the specified timezone and locale.
date(timezone, locale, year, month, day, hour, minute, second)	A date with the specified year, month, and day and time including seconds, in the specified timezone and locale.
date(timezone, locale, date)	A date with the specified date value in the specified timezone and locale.
time(timezone, locale, hour, minute)	A time with the specified hour and minute (hours are specified as a 24 hour clock), in the specified timezone and locale.
time(timezone, locale, hour, minute, second)	A time with the specified hour and minute and second (hours are specified as a 24 hour clock), in the specified timezone and locale.

The `timezone` and `locale` parameters are strings that are timezone or locale designators. Either can be `null`.

- `timezone` is a full name, such as "America/Los_Angeles", or a custom ID such as "GMT-8:00" (see `java.util.TimeZone.getTimeZone(id)`). The GMT zone is used if the given ID cannot be understood.
- `locale` can be a language code alone, or a language code and country code separated by "-", as in "en-us".

The rest of the parameters used in these methods are as follows:

- `year` - an integer specifying the year (for example, 2009).
- `month` - an integer specifying the month with a value ranging from 1 to 12. The `getMonth` date method returns a value of 0 to 11 so don't forget to add 1 if constructing a new date from an existing date using the `getMonth` method.
- `code day` - an integer specifying the day of the month with a value ranging from 1 to 31.
- `hour` - an integer specifying the hour of the day with a value ranging from 0 to 23.
- `minute` - an integer specifying the minute of the hour with a value ranging from 0 to 59.
- `second` - an integer specifying the second of the minute with a value ranging from 0 to 59.

The current date and time is always available using the `now` operator. The current date is always available using the `today` operator. The expression package provides a large number of date operators, including relative date arithmetic operators as in the following example (which returns the 2nd monday of next month):

```
now + 1 end of months + 2 mondays
```

The available time units for the add and subtract time unit operators are as follows:

- minutes
- hours
- days
- weeks
- months
- years
- mondays
- tuesdays
- wednesdays
- thursdays
- fridays
- saturdays
- sundays
- end of months
- end of years

Date methods

Method	Returns
<code>getDayOfMonth()</code>	integer, the day of the month (between 1 and 31)
<code>getDayOfWeek()</code>	integer, the day of the week (1 = Sunday, 2 = Monday, 3 = Tuesday, 4 = Wednesday, 5 = Thursday, 6 = Friday, 7 = Saturday)
<code>getDayOfYear()</code>	integer, the day number within the year; the first day of the year is 1
<code>getHour</code>	integer, the date's hour of the morning or afternoon (used for 12-hour clock)
<code>getHourOfDay()</code>	integer, the hour (24 hour clock, between 0 and 23)
<code>getMinute()</code>	integer, the minute within the hour (between 0 and 59)
<code>getSecond()</code>	integer, the second within the minute (between 0 and 59)

Method	Returns
getMonth()	integer, the date's month (January = 0, December = 11)
getYear()	integer, the year (for example, 2005)

Examples

The `at` operator enables you to copy the time from one date variable into another:

```
item.Properties.TheDate at item.Properties.MyDateTime
```

If you want to do something special on a Friday, you could use the following expression syntax:

```
now.getDayOfWeek() == 5
```

Notes

- A Date variable always includes a time, even if it is created with just a date; if the time is not specified it is set to midnight - 12:00am.
- A Date variable always includes a date, even if it is created with just a time; if the date is not specified it is set to 1/1/1970.

String methods

A number of built-in string methods are provided to work with strings. The following sample illustrates the use of the string method `split` to break portions of a string separated by semicolons into an array of strings.

```
"foo;bar;widget".split(";;")
```

The following table provides the complete list of built in string methods.

Method	Arguments	Type	Returns
compareToIgnoreCase(string)	string	string	integer, negative, zero, or a positive as the specified string is greater than, equal to, or less than this string, ignoring case considerations
endsWith(suffix)	suffix	string	boolean, true if the string ends with the specified suffix
equals(string)	string	string	boolean, true if the two strings are

Method	Arguments	Type	Returns
			exactly equal (this is the same as the == operator)
equalsIgnoreCase (string)	string	string	boolean, true if the two strings are equal ignoring case
indexOf (substring)	substring	string	integer, the index within the string of the first occurrence of the specified substring
indexOf (string, offset)	substring string	offset integer	integer, the index within the string of the first occurrence of the specified substring after the specified offset
lastIndexOf (substring)	substring	string	integer, the index within the string of the last occurrence of the specified substring
lastIndexOf (substring, offset)	substring string	offset integer	integer, the index within the string of the last occurrence of the specified substring, searching backward starting at the specified offset
length ()			integer, length of the string
matches (regex)	regex	string	boolean, true if the string matches the given regular expression (see java.util.regex.Pattern for details)
replaceAll (regex, replace)	regex replace	string string	string, with each substring of the string matching the given regular expression replaced with the given replacement
replaceFirst (regex, replace)	regex replace	string string	string, with the first substring of the string matching the given regular expression replaced with the given replacement
split (regex)	regex	string	string list, containing each substring of the string that is terminated by another substring matching the given regex or terminated by the end of the string. The substrings in the list are in the order in which they occur in the string. If the expression does not match any part of the input then the resulting array has just one element, the original string

Method	Arguments	Type	Returns
split(regex, limit)	regex limit	string integer	Same as above, except with a result threshold (see java.lang.String split method)
startsWith(prefix)	prefix	string	boolean, true if the string starts with the specified prefix
startsWith(prefix, offset)			boolean, true if the substring starting at the given offset starts with the specified prefix.
substring(begin)	begin	integer	string, the substring from the beginning index (inclusive) to the end of the string
substring(begin, end)	begin end	integer integer	string, the substring from the beginning index (inclusive) to the end index minus 1
toLowerCase()			string, the string with every character forced to lower case
toUpperCase()			string, the string with every character forced to upper case
trim()			string, with leading and trailing whitespace removed

Arrays and aggregation

Array data types are a natural consequence of relationships (both peer-to-peer "to many" or child relationships). The expression engine provides various ways to interact with arrays:

```
array {"a", "b", "c", "d"}  
array1.size()  
array1.indexOf("b")  
array1.lastIndexOf("c")  
array1.contains("b")  
array1.isEmpty()  
array1.get(2)
```

Arrays can be referenced using normal array syntax. For example:

```
array1 = array{"a", "b", "c", "d"}  
s1 = array1[2]  
s2 = array1[7]  


- s1 equals the string "c"
- s2 equals null - when an index value is out-of-bounds a null value is returned.

```

Array processing

The expression engine provides the ability to operate on arrays using the empty array syntax [] followed by dotted notation to indicate an operation is to be performed on every entry in an array. These operations result in new arrays the same length as the original array. The following example applies the substring method to every string in an array to produce a new array of strings.

```
array1 = array("sitting", "history", "coke")
array2= array1[].substring(1,3)
■ array2 equals the array {"it", "is", "ok"}
```

Following is another array processing example, using an item that has a child entity named line with two properties named quantity and price.

```
result = line[].quantity * line[].price > 100
■ result is a boolean array indicating whether each related row's quantity times price is greater than 100
```

Most operators support array processing. To be more precise, scalar operators can be used with arrays as long as the operator is defined with one or more scalar operands (even if other operands are non-scalar), and the operator returns a scalar value. For these operators, any one or more of the scalar operands can be used with array arguments, in which case the resulting calculation causes iteration across the array(s), and produces an array as the result.

The following example shows an array being passed into a method that normally takes a scalar argument (the method contains returns true if the array includes an entry that matches the argument).

```
myContents = array{ "Form1022", "PoliceReport", "A54" }
array1 = contents.contains(myContents)
■ array1 is a boolean array, length 3, indicating whether each of the three contents are in the string mydata
```

The following example is more involved. The inner clause contents.get(myContents) returns an array of items (instances of the contents child entity) specified by the array of names. This array is then processed to return an array that contains only the status property from each of those items.

```
myContents = array{ "Form1022", "PoliceReport", "A54" }
statusArray = (contents.get(myContents)) [].status
■ statusArray equals an integer array, length 3, of the status code of the three desired items from the contents child entity.
```

Arrays and basic operators

The expression package provides automatic array processing across array members, whenever an array is used as an argument to an operator that normally performs calculations only on scalars. Consider the following example of an operator that normally works with scalar values.

```
a = 5;  
b = 7;  
c = a + b;  
■ c equals the integer 12
```

The same operator can be used with an array and a scalar value, returning an array value:

```
a = array{ 10, 20, 30, 40, 50 };  
b = 4;  
c = a + b;  
■ c equals the integer array { 14, 24, 34, 44, 54 }
```

The same operator can be used with two arrays (nulls are treated in the same way as in scalar operations):

```
a = array{ 10, 20, 30, 40, 50 };  
b = array{ 1, 2, null, 4, 5 };  
c = a + b;  
■ c equals the integer array { 11, 22, null, 44, 55 }
```

The same operator can be used with two arrays of different lengths. The smaller array is automatically padded with nulls.

```
a = array{ 5, 10, 8, 4 }  
b = array{ 9, 5, 12, 3, 7, 2 }  
c = a + b;  
■ c equals the integer array { 14, 15, 20, 7, null, null }
```

Array object types

The only source of arrays is the items referenced by peer-to-peer to many or child relationships.

Arrays of items can be referenced using the array syntax described previously. For example, the date time a task was created, for the first task item in a case:

```
item.tasks[0].properties.dateTimeCreated
```

The value of the amount property of the fifth item in the set of related items identified by the relation customer (since ordering is not specified, this is not very useful):

```
item.customer[4].properties.amount
```

The entire set of task items in a case entity can be referenced using the empty array syntax.

```
item.tasks[]
```

An array of the task's deadline times can be extracted as an array of date/time values:

```
item.tasks[].deadline.time
```

An array of the amount values for the line items in an invoice can be extracted into an array of decimal values:

```
item.line[].properties.amount
```

The following example shows how the qualified name can include a method call, which is applied to every member of the array. This expression returns an array of integer values representing the number of versions in each document referenced by a case:

```
item.references[].item.versions.size()
```

Aggregation operators

Aggregation operators are operators that take an array of values, do some calculation across the entire array, and (usually) return a scalar value. For example:

```
int x = sum( item.lines[].properties.amount )
```

When doing calculations, null (unknown) values may be important in some situations, or unimportant (should be ignored) in others. Each aggregation operator has two variations, one that ignores the null array members (as if they were not in the array), and one that pays attention to nulls (which for most aggregation operators, means the entire result is null, but there are some aggregation operators that have more intelligence).

The default behavior for aggregation operators is to ignore null members. To pay attention to nulls, you use a different operator name with the suffix "N" ("N" means null-aware):

```
int x = sumN( lines[].amount )
```

Examples:

```
array1 = array{ 5, 9, 3, null, 5 };
s1 = sum(array1);
s2 = sumN(array1);
■ s1 equals the integer 22
■ s2 equals null
```

Handling uncertainty

`countTrueN` and `percentTrueN`, the null aware versions of two aggregation operators, exhibit more intelligence than most operators in that they retain knowledge about possible outcomes by calculating their result as a range data type.

```
array1 = array{ true, true, false, null, true, null, false, true }
c = countTrueN(array1);
p = percentTrueN(array1);
```

- c equals the range {4, 6}
- p equals the range {50, 75}

The ranges indicate the range of possible results once the currently undefined null values are set to either true or false. At this time, 4 out of the 8 values are true, but 2 are null, so that it is still possible, once those 2 other values are known, that 6 values may be true". The final result is the possible range. These two aggregation operators retain as much knowledge about unknown values, so that other operators can use the information to still (possibly) calculate definitive results. For example, the range value can then be compared against a scalar value, which may return a definitive value as follows:

```
b1 = ( c >= 4 );  
b2 = ( p >= 50 );
```

- Both b1 and b2 are true (enough information is known to definitely reach a result).

However, the comparison of a range to a scalar value may sometimes result in a null (unknown) value if there is not enough information to produce a definitive answer:

```
b3 = ( c > 5 );  
b4 = ( p > 60 );
```

- Both b3 and b4 are null (not enough information is known to determine a result).

This type of logic is useful in process logic, as in a gateway with the following condition to route if more than 50 percent of the voters approve a decision:

```
percentTrueN(votes[] .approve) > 50
```

The operators ==, !=, >, >=, <, <= support comparisons between a range and scalar value, returning a Boolean (true/false if there is enough information to know the result, or null if not enough information is known). The operators +, -, *, /, and - (negation) support ranges (accept ranges as operands), and propagate known information by returning a range value as its result.

The aggregation operators

There are two versions of each aggregation operator, one that ignores null values and one that does not. The null aware version of an aggregate operator is suffixed with the letter N. The aggregation operators are described in the following table.

Operator	Description
countTrue	Returns the number of true values in the array (only for Boolean arrays). The null aware version returns a range of possible true counts.
percentTrue	Returns the percentage (an integer from 0 to 100) of true values in the array (only for Boolean lists). The null aware version returns a range of possible true percentages.
sum	Returns the sum of all values in the array (for arrays of integer, long, or decimal values). The null aware version returns null if any value in the

Operator	Description
	array is null.
average	Returns the average of all values in the array (for lists of integer, long, decimal, or date values). The null aware version returns null if any value in the array is null.
median	Returns the median value in the array. If the array has an even number of entries, it returns the average of the two middle values (for arrays of integer, long, decimal or date values). The null aware version returns null if any value in the array is null.
mode	Returns the mode (the most frequently occurring value) in the array (for arrays of integer, long, decimal, date, or string values). If two values occur the same number of times, it returns the value that occurs first in a sorted array. If no value occurs more than once (all equal), it returns null. The null aware version may return a value even if nulls exist in the array, if the leading candidate has an overwhelming lead (if it occurs more frequently than the next leading candidate plus the number of null values).
min	Returns the lowest value in the array (for arrays of integer, log, decimal, or date values).
max	Returns the highest value in the array (for arrays of integer, log, decimal, or date values).

Chapter 12

Administering an application

Use the Process Experience Administration tool to perform administrative functions for your applications.

Note: Applications are called solutions in the Administration tool. Solutions can be applications installed with the base product, deployed using Application Deployer, or projects published from CWS Workspace.

To run the Administration tool, type the following address in your browser's address box:

`https://<server>:<port>/home/<organization>/app/admin/web/config`

Replace the variables <shown in angle brackets> with actual values for your installation, as shown in the following example:

`https://localhost:8080/home/system/app/admin/web/config`

To run the Administration tool, a user must have the **Manage Solution Security** permission. Use the Process Platform User Manager to select the appropriate roles (you need to select Include internal roles to see the additional roles).

Note: You must have the **Entity Runtime Administrator** role to access the Administration tool to set up the initial permissions and roles.

Defining security on a solution

When solutions (applications) are deployed or published, they are closed by default. To enable a solution in Process Experience, assign the user to the **Entity Runtime User** role or to any other role and then use the Process Experience Administration tool to add the role to the **Use Solution** permission that is located in the Solution Security section.

Solution security policy

A solution has a security policy that controls access to the solution. The security policy is accessed in the Administration tool. The permissions exposed in the security policy control access to the solution and are described as follows:

- **Use Solution** enables using the solution. A user cannot access items in solutions to which they do not have Use Solution permission.

- **Administer Solution** enables administration of the solution. This enables getting and setting the solution's configuration properties. There is no way to enabling administering some configuration properties in a solution and not others.
- **Manage Solution** enables managing a configuration. This includes the ability to modify the availability of versions of the solution.
- **Manage Solution Security** enables managing a solution as a whole. This includes changing the solution security policy.
- **Build** enables modifying the solution's definition. This includes the ability to delete elements.
- **Include** enables the inclusion of the solution in another solution. After a solution has been included in another solution, anyone with Build permission to that solution will see everything in the included solution (even if they do not have Build or Include permission to the included solution).

Solution creation

Process Experience comes with three predefined roles:

- Entity Runtime Administrator
- Entity Runtime Developer
- Entity Runtime User

These roles are used to set the initial security of a solution. When a solution is created, roles are assigned to permissions as shown in the following table.

Permission	Role
Administer Solution	Entity Runtime Administrator
Build Entity	Entity Runtime Developer
Include Entity	Entity Runtime Developer
Manage Solution	Entity Runtime Administrator
Manage Solution Security	Entity Runtime Administrator
Use Solution	

Since no roles initially have the Use Solution permission, the application is not usable by a user until an administrator grants access.

If you are a solution designer, you should become a member of the Entity Runtime Administrator role and the Entity Runtime Developer role.

Configuring solution security

The Administration tool is used to configure security. Only users with the **Manage Solution Security** permission can configure security.

Before you begin:

- [Create a workspace and project.](#)
- [Create at least one entity in the project.](#)
- [Publish your project.](#)

To configure solution security:

1. Run the Administration tool by typing the following address in your browser's address box:

`https://<server>:<port>/home/<organization>/app/admin/web/config`

Note: Replace the variables <shown in angle brackets> with the relevant values for your installation. For example:

`https://localhost:8080/home/system/app/admin/web/config`

2. In the **Workspaces** list in the left pane, select your solution, for example **OpenTextOrderManagement**.

The workspace name contains the name of the Package Owner followed by the Product Name that you entered when you created the workspace in CWS.

3. In the **Configurable Elements** list in the middle pane, expand **Solution Security** and select **Use Solution**.
4. In the **Roles** list in the right pane, select the following roles:
 - Entity Runtime Administrator of OpenText Entity Runtime
 - Entity Runtime User of OpenText Entity Runtime
 - Entity Runtime Developer of OpenText Entity Runtime

Defining security for Inbox Task Management

When you install Process Suite, a shared application called Inbox Task Management is automatically installed. The Inbox Task feature displays items from a user's Process Platform Inbox in a Process Experience list. See [Using Inbox lists](#).

Note: Inbox Task Management does not support notifications used in the MyInBox Explorer.

To make the Inbox lists available in Process Experience, you need to set solution security for the Inbox Task Management application.

To set solution security for the Inbox Task Management application:

1. Open the Process Experience Administration tool.
2. On the right side of the header area, select **Global** in the drop-down list to show your workspace.
3. In Workspaces, select **OpenTextInboxTaskManagement**.
4. In Configurable Elements, select **Solution Security > Use Solution**.

5. In Roles select the following roles:
 - Entity Runtime Administrator of OpenText Entity Runtime
 - Entity Runtime User of OpenText Entity Runtime
 - Entity Runtime Developer of OpenText Entity Runtime

Note: Your changes are saved automatically. You do not need to click **Save** or **OK**.

Properties added

The Inbox Task Management application adds several properties to the Task child entity of the Lifecycle building block. You can use these properties in building blocks, such as forms and worklists, that you add to the Task entity. See [Configuring the task entity > Task properties](#).

Defining security for the Identity package

When you install Process Suite, a shared application called Identity package is automatically installed. This feature enables an administrator to manage users in Process Experience.

Solution level security is managed out of the box during Identity package deployment. However, user level security, if needed, should be managed by application administrator. See the Process Experience documentation for details.

The following are the roles and corresponding features in Identity package.

Name	Description
Identity Administrator	Subrole of Administrator. Allows managing of user information from the Identity Homepage in Process Experience.
Identity Push Service	Subrole of OTDS Push Service. Needed on OTDS Resource User configuration for synchronization.
Identity User	Allows access to the Inbox.

Defining solution variables

You can use the Administrator tool to define solution variables that represent values that are subject to change, such as a discount rate. One possible use for a solution variable could be as a part of the URL in a web content panel. The Name of a solution variable must exactly match the name of the element that it represents. For example if the variable represents a property called Discount, it must be named Discount.

To define solution variables:

1. In the Solutions panel, select the solution where the variable will be used.
Each variable for that solution is listed in the Configuration Properties panel.

2. Select one of the following options:
 - To create a variable, click **Create**, enter a Name and Value, and then click **Add**.
 - To modify a variable, type the name or value over the existing entries.
 - To delete a variable, click its Delete icon.

Changing the availability of a solution

When a solution is disabled or made unavailable, neither home pages nor lists are available to a user. Typically, a solution is temporarily disabled in the entity run time to perform database maintenance.

Only users with **Manage Solution** permission can change the availability of a solution.

To change the availability of a solution:

1. In the Solutions panel, select the solution.
2. In the Solution Configuration panel, select **Solution Status > Availability**, and then specify the availability.

Displaying a solution's errors and warnings

When a solution is in an error state it is automatically disabled and all the lists and create menu entries disappear in Process Experience. By viewing a solution's error list, you can troubleshoot why it disappeared.

Only users with **Build** permission can view a solution's errors and warnings.

To display a solution's errors and warnings:

1. In the Solutions panel, select the solution.
2. In the Configurable Elements pane, select **Solution Status > Error and Warnings**.
The Errors and Warnings pane displays any errors or warnings.

Configuring security for layouts

By default, all users who are members of the Entity Runtime User role have access to all layouts. Using the Administration tool, you can associate specific layouts in a solution with Process Platform roles. This provides an additional layer of security to that provided by the Security building block.

To configure security for layouts:

1. In the Solutions panel, select a solution.
The layouts and variables for that solution are listed in the Configurable Elements panel.
2. In the Configurable Elements panel, expand Layout Security and select the layout for

which you want to assign roles.

3. In the Roles panel, select the roles to assign to the selected layout.

Caution: Remove the **Entity Runtime User** role from a layout only after you add a different role to it. Otherwise, it becomes inaccessible.

Deleting a solution

Solutions listed in the Process Experience Administration Workspaces column include solutions provided with the base product, applications deployed by Application Deployer, and solutions created by publishing projects from the CWS workspace. This topic describes deleting solutions created by publishing projects from the CWS workspace.

Important: Application Deployer installed solutions should only be removed using the Application Deployer undeploy feature.

You might need to delete a solution from Process Experience when you delete a top level model such as an application home page or entity. When you delete a solution, any data that you entered in Process Experience is deleted but your project remains intact in the CWS workspace and can be republished to Process Experience.

Only users with **Build** permission can delete a workspace.

To delete a solution:

1. In the **Workspaces** panel, select the solution to delete.
2. Click the gear icon and select **Delete**.
3. Click **OK** when prompted for confirmation.

Deleting an entity

This option is provided to delete an entity created by publishing a project from the CWS workspace during application development.

You might need to delete an entity when you remove it from your project in CWS and then want to delete the entity from the Process Experience solution.

Only users with **Build** permission can delete an entity.

To delete an entity:

1. In the **Workspaces** panel, select the solution and then select the entity in the list of entities.
2. Click the gear icon and select **Delete**.
3. Click **OK** when prompted for confirmation.

Publishing a solution

You can publish an entity directly from the Process Experience Administration tool.

To publish a solution:

1. In the Workspaces panel, select the solution.
2. Click the gear icon and select **Publish**.

Deleting an EIS entity

This option is provided to delete an EIS entity created by publishing a project from the CWS workspace during application development.

You might need to delete an EIS entity when you remove it from your project in the CWS workspace and then want to delete the entity from the Process Experience solution.

Only users with **Build** permission can delete an EIS entity.

To delete an EIS entity:

1. In the **Workspaces** panel, select the solution and then select the entity in the list of EIS entities.
2. Click the gear icon and select **Delete**.
3. Click **OK** when prompted for confirmation.

Deleting history logs

This option is provided to delete a history log created by publishing a project in the CWS workspace during application development.

You might need to delete a history log when you remove it from your project in the CWS workspace and then want to delete the history log from the Process Experience solution.

Only users with **Build** permission can delete a history log.

To delete a history log:

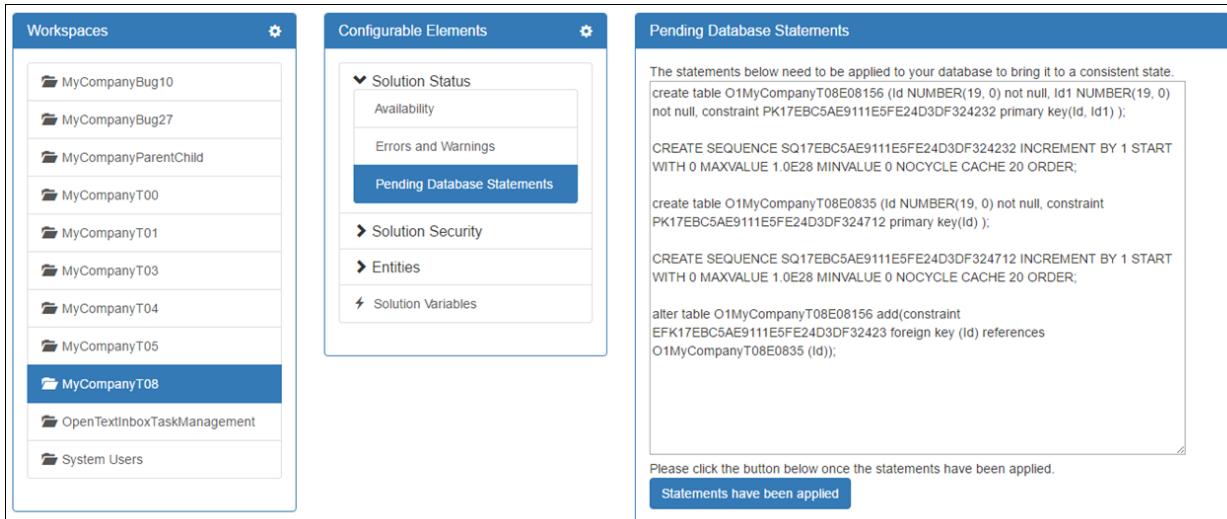
1. In the **Workspaces** panel, select the solution, and then select the history log under history logs.
2. Click the gear icon and select **Delete**.
3. Click **OK** when prompted for confirmation.

Applying missing DDL statements

Executing Database Definition Language (DDL) statements in Oracle has the side effect of committing transactions. If DDL statements are intermixed with Database Manipulation Language (DML) statements and something goes wrong, the affected solution typically ends up in an inconsistent state. To avoid this problem, all the DML statements are executed before any DDL statements are executed so a failure in DDL processing commits all DML statements but leaves the system in an internally consistent state, minus the failed DDL

statements. So while the solution is internally consistent, it is not possible to create items until the database reflects the failed DDL statements.

The failed DDL statements are recorded and made available to the database administrator in the Administrator tool so that they can be applied to the database. When a solution has pending DDL statements, a new option is displayed in the Solution Status section, as shown in the following illustration.



The database administrator can take the statements shown in the right panel and apply them to the database and, once they are applied, click the button. This tells the system to calculate the data model again. If there are still statements pending (they were not all applied to the database) they appear in the right panel again. Once the statements are applied and the button clicked, the right panel is cleared.

Selecting a database configuration

A solution can have one or more database references. When a solution has one or more database references, the Configurable Elements panel contains a Database References accordion pleat. This pleat shows all available database references in the solution. You select the database configuration to use by selecting a database reference.

When you select a database reference, the Database Reference panel (on the right) shows the associated database configurations. You can then select the database configuration you want from the list. The list is grouped by organization and shows all configurations available in your organization as well as those available in the system organization. It also shows where the database reference is used and lists all entities in all solutions that use it.

When a database reference is created, there is no associated database configuration set yet and the solution's status displays a warning alerting you to this. When you open that database reference, you are prompted to select a database configuration.

Note: In **Configurable Elements > Entities**, if you click an entity that is associated with a database reference, the Entity Properties panel (on the right) shows the database reference it uses.

Specifying the email configuration

The first time you publish a project that contains the Email building block, you must use the Process Experience Administration tool to populate the information for the email configuration. If you fail to do this, emails cannot be sent.

Thereafter, you can use the Process Experience Administration tool to select an email configuration and provide appropriate values to the fields. For example, you may need to provide From Mail-ID, Display Name, and so forth.

To add or modify an email configuration:

1. In the Process Experience Administration tool, select your workspace in the Workspaces list.
2. In **Configurable Elements**, select the e-mail configuration that you want to change.
3. In **Configuration Properties**, define the following values:
 - a. In **From**, provide the email address from which emails will be sent.
 - b. In **Display Name**, type the display name of the user with whose email ID the emails will be sent.
 - c. Select **Use the full name of the logged in user as the display name**. The **Display Name** will be empty and the logged in user's full name will be used as the display name.
 - d. In **Reply to**, type the email address to which replies will be sent.
 - e. In **Email profile user ID**, provide the user's profile user ID, which can be obtained from Process Platform User Management. For example, jsmith.

This user's credentials will be used to access the mail server and to send email. To configure the authentication for a user ID, invoke the [SetProfile web service](#). The setProfile web service must be invoked from the same organization that the application is deployed on. This is mandatory if the email server requires authentication.

Configuring Extended ECM for Process Platform

This section explains how to configure Extended ECM for Process Platform.

Configuring a business workspace in Process Platform

The basic steps to create and open a business workspace in Process Platform are as follows:

1. Configuring a business workspace in Content Server.
2. Creating a document store application connector.
3. Opening the business workspace in Process Experience.

Configuring a business workspace in Content Server

To create a business workspace in Content Server from Process Platform, configure Content Server as described in the *OpenText Extended ECM Platform Integration Guide*.

The high level configuration steps are as follows:

1. Connecting an external system. See [Connecting an external system](#)
2. Creating a category for the workspace type and the business object type. See [Creating a category](#)
3. Creating a classification for document templates.
4. Creating a location for the business workspaces.
5. Creating a workspace type.
6. Defining a document template for business workspaces.
7. Configuring business object types.

Connecting an external system

To establish a connection from Content Server to Process Platform:

1. On the Content Server Administration page, click **Extended ECM**.
The configuration steps are available on the right side of the page.
2. Click **Configure Connections to External Systems > Add External Systems**.
3. Provide the following application server details.

Option	Description
Logical External System Name	Provide a name. This name will be used in the External System Id field while configuring the document store connector for Process Platform.
Connection Type	Displays the connection type.
Enabled	Select this option to enable the configuration.
Comment	Type a comment to provide additional information.
Base URL	Common URL for accessing applications through a Web browser. You can use this base URL when configuring business object types on Content Server. For example: http://<server>:<port>
Application Server Endpoint	Specify the URL that will be called to obtain business object information. For example, for Process Platform the URL is: http://<server>:<port>/home/<organization>/app/xecm/ECMLinkService?WSDL
Schema	Select the interface version.

Option	Description
Version	
User Name	Enter the user that is used to access business object type information from the external system.
Password	Specify the password of the defined user.
Test Connection	Click Test to perform a connection check to the specified external system. After a successful check, the message Connection test to <Logical External System Name> was successful is displayed next to the button. The system ID is also retrieved and displayed.

Note: The external system must be configured for each organization separately

Creating a category

Custom categories will help you to add relevant metadata to business workspaces or documents.

Categories and attributes can be used for:

- Workspace type configuration.
- Business object type configuration where entity properties can be mapped to category attributes.

The following table shows the datatype mappings between Process Platform entity properties and Content Server category attributes.

Process Platform	Content Server
Boolean Checkbox	Flag: Checkbox
Date	Date: Field
DateTime	Date: Field Note: Select Include Time Field during attribute creation.
Decimal	Text: Field
Duration	Text: Field or Text: MultiLine
Integer	Text: Integer
Float	Text: Field
Enumerated Integer	Integer: Popup
Long Text	Text: MultiLine
Text	Text: Field
Enumerated Text	Text: Popup
Currency	Text: Field

Note: The Set and Partner Set category attributes of Content Server are supported in Process Platform for mapping entity properties.

Creating a document store application connector

To create a document store application connector for Content Server, see the page called *Creating and Modifying Document Store Repository* in the Process Platform Product documentation.

Note: If **Synchronize Workspaces Manually** is selected in the document store application connector configuration, a Process Experience user sees a **Synchronize workspace** button in the action bar for the Results panel. The user must click this button to create a business workspace in Content Server. If **Synchronize Workspaces Manually** is cleared, the business workspace is automatically created in Content Server when an entity instance is created by the application.

Opening the business workspace in Process Experience

To open a business workspace from Process Experience, a user must be added to both Process Platform and Content Server.

To open the business workspace in Process Experience:

1. On the Content Server Administration page, select **Server Configuration > Configure Security Parameters**.
2. Add the following application server details:
 - Trusted Referring Websites
 - Trusted Cross Domains
 - Frame Embedding: Clear **Prevent request handlers from being embedded in external frames**.
3. Save the configuration.

Business Workspace Configuration Report

As an administrator, you can view the report that contains information about various configurations required for Extended ECM (xECM) and a list of the entities that are configured for xECM.

The following reports are available:

- The **Document Store Configuration Report** displays details of the document store connector configuration available in the current organization.
- The **Administrator Configuration Report** displays the business workspace configuration done for entities using the Process Experience administration interface. Entities are shown in a list. Select an entity to view the business workspace configuration.

- The **Application Configuration Report** displays the entities that are configured with a business workspace building block and their properties. Entities are shown in a list. Select an entity to view its properties and their corresponding data types.
- The **Content Server Configuration Report** displays the business object type configuration in content server for the selected entity. See [To view a Content Server configuration in Process Platform](#).

To view a business workspace configuration report in Process Platform:

1. Start the Process Experience Administration tool.
2. In Workspaces, select the project where the entity is configured.
3. In Configurable Elements, click **Business Workspace Configuration Report**.
The Configuration Properties panel displays a list of reports.

Note: A green check mark indicates that no action is needed. A warning symbol indicates that you should ensure that the value given in the configuration is consistent with the product documentation.

4. Expand each option to view the related report.

To view a Content Server configuration in Process Platform:

1. Export the Content Server configuration.
 - a. In Content Server, Go to Content Server > Admin > navigate to Connected Workspaces and click Import/Export Configuration.
 - b. Navigate to Export Configuration, provide the exported file name, select the relevant business objects, and click Export.
The file is created and the file properties are opened.
2. Upload the Content Server Configuration file in Process Platform.
 - a. Navigate to the Process Experience Administration tool > project where the entity is configured > **Business Workspace Configuration Report** > **Content Server Configuration Report**.
 - b. Using the **Browse** button, select the exported file that you downloaded.
 - c. After the file is uploaded successfully, select an entity to see the business object type configuration for it.

Note: If the file was exported in Content Server 16.2 or greater, only details about business object types are displayed. For Content Server versions earlier than 16.2, you can view details for both business object types and workspace types.

Configuring Content Server to work with business attachments in Process Platform

To configure Content Server to work with business attachments in Process Platform:

1. On the Content Server Administration page, click **Extended ECM**.
The configuration steps are displayed.
2. Click **Configure Business Object Types**.
The Configure Business Object Types page opens with the available list.
3. Click the business object type for which a business attachment needs to be configured.
The Business object type configuration window opens.
4. Specify the following properties:
 - **Can be Added as Business Object** - Select **Yes**.
 - **Business Object Name Pattern** - Provide a name pattern with which business attachments will be created in Content Server from Process Platform.
For example, [Properties.EntityProperty] creates business attachments with the EntityProperty name.
5. Click **Save Changes**.

Important: In Content Server, Process Platform users should have permission to use documents as attachments in Process Experience.

Configuring a cross application business workspace

If you have similar business object types in different applications, such as a customer in a Contract Management application and a business partner in a Customer Management application, you can create one cross application business workspace for two or more business objects of different types and from different applications. This enables sharing a business workspace with multiple applications. It also enables business workspaces of different applications to be related to each other so that the metadata of one application can be viewed in another application.

In scenarios where a business workspace needs to be shared with multiple applications, one of the applications becomes a leading system. For example, assume that a leading application has business workspaces created in Content Server. An application in Process Platform that needs to share or relate to a Content Server business workspace needs to configure the following settings in the Process Experience Administration tool

The following sections explain how to configure a cross application business workspace.

To configure Content Server for cross application support in Process Platform:

1. Go to **Content Server > Admin tab > Content Server Administration > System Object Volume > Enterprise Data Source Folder > Enterprise Search Manager > Properties > Regions**.

2. Set the following properties:
 - Select **Queryable** for XECMWkspLinkRefTypeID.
 - Select **Displayable** for XECMWkspLinkSAPObjectKey.
3. Click **Update** to save the configuration.

To configure Process Platform for cross application business support:

1. Start the Process Experience Administration tool.
2. In **Workspaces**, select the project where the entity is configured.
3. In **Configurable Elements > Entities**, select the entity that needs to share or relate to a leading business workspace.
4. In **Extended ECM**, provide the following configurable parameters.
 - a. In Cross Application Business Workspace, select **Shared** or **Related**.
To get this value, Go to **Content Server > Admin tab > Content Server Administration > Extended ECM > Configure Connections to External Systems** and select the **Logical External System Name** that is configured to the leading system.
 - b. In External System ID, provide the External System ID of the leading system configured in Content Server.
To get this value, go to **Content Server > Enterprise > Connected Workspaces > Workspace Types**. Click the leading system's Workspace Type to open it. The URL in the browser's address field now shows a string that contains the parameter ID_CFG, for example, ReferenceTypeEdit&ID_CFG=24. Make a note of the value, in this example, 24.
 - c. In Workspace Type ID, provide the workspace id of the leading system configured in Content Server.
To get this value, go to **Content Server > Enterprise > Connected Workspaces > Workspace Types**. Click the leading system's Workspace Type to open it. The URL in the browser's address field now shows a string that contains the parameter ID_CFG, for example, ReferenceTypeEdit&ID_CFG=24. Make a note of the value, in this example, 24.
 - d. In Business Object Type, provide the business object type of the leading system configured in Content Server.
To get this value, Go to **Content Server > Admin tab > Content Server Administration > Extended ECM > Configure Business Object Types**. Click the leading system's Business Object Type and select the Business Object Type from the Configure Business Object Type page.
 - e. In Search Expression, provide the search expression to use to search the business workspace in the leading system.
To get this value, Go to **Content Server > Enterprise > Connected Workspaces > Workspace Types**. Click the leading system's Workspace Type and read the pattern from Business Workspace Names. For example, if in Content Server the pattern is [100331:Attribute1]-[100331:Attribute2], the search expression should be as follows: item.Properties.EntityProperty1+ "-" + item.Properties.EntityProperty2. Using this search expression, available business workspaces will be listed in Process Experience to link to an entity instance.
 - f. If the application needs to pull data from the shared workspace to the entity instance, in **Map entity properties to Content Server categories**, map the entity properties with the Content Server category attributes of the leading system

for which you want to view Content Server data in Process Platform. This is an optional configuration and is applicable for cross application **shared** business workspaces.

- In Select an Entity Property, select an entity property.
- Click the **Select Category Attribute** search button.
The Select a Category window opens.
- Select the leading system category and click **Select**.
- Click **Select Category Attribute** to list the attributes of the selected category.
- Select a required attribute and click **Add** to define the mapping.
You can define multiple mappings.

You can delete a mapping by selecting it and clicking **Remove**.

Configuring a link to an existing business workspace

In many scenarios, Content Server will be the user-facing application and users will be managing workspaces and documents directly using Content Server without using any other applications. Therefore, users will be creating workspaces and providing metadata using Content Server. Other business applications need to reuse this already existing workspace. This configuration enables a Process Experience user to link to Content Server business workspaces from entity items.

Note: The option to link an existing business workspace is available only if Process Platform is configured with Content Server 16.2 or greater.

The following sections explain how to configure a link to an existing business workspace:

- [Configuring Content Server for Link business workspace support](#)
- [Configuring a link to an existing business workspace in Process Platform](#)

Configuring Content Server for Link business workspace support

To configure Content Server to link business workspace support in Process Platform:

1. Go to **Content Server > tools > facets volume >workspace columns >workspace name en_us > function menu >properties > workspaces**.
2. Select **Used for Sorting and Filtering**.
3. Click **Update** to save the configuration.

Configuring a link to an existing business workspace in Process Platform

This configuration enables a Process Experience user to link to Content Server business workspaces from entity items.

To configure a link to an existing business workspace:

1. Start the Process Experience Administration tool.
2. In **Workspaces**, select the project where the entity is configured.
3. In **Configurable Elements > Entities**, select the entities to link to an existing business workspace.
4. In **Extended ECM**, provide the following configurable parameters.

- a. In **Business Workspace Configuration**, select **Link Existing Business Workspace**.
- b. In **Workspace Type ID**, provide the workspace ID of the entity configured in Content Server.

To get this value, Go to **Content Server > Enterprise > Connected Workspaces > Workspace Types**. Click the configured system's Workspace Type to open it. The URL in the browser's address field shows a string that contains the parameter ID_CFG, for example, ReferenceTypeEdit&ID_CFG=24. Make a note of the value, in this example 24.

- c. In **Search Expression**, provide the search expression to use to search the existing business workspace.

To get this value, Go to **Content Server > Enterprise > Connected Workspaces > Workspace Types**. Click the configured Workspace Type and read the pattern from Business Workspace Names. For example, if in Content Server the pattern is [100331:Attribute1]-[100331:Attribute2], the search expression should be as follows: item.Properties.EntityProperty1+ "-" + item.Properties.EntityProperty2

- If automatic workspace creation is enabled, this search expression is used to find the workspace that needs to be linked during entity instance creation.
 - If manual workspace creation is enabled, the search expression is not used and, instead, a Content Server widget is used to search and list all workspace instances.
- d. If the application needs to pull data from the Content Server workspace to the entity instance, **Map entity properties to Content Server categories** helps to map the entity properties with the Content Server category attributes for which you want to pull data from Content Server.
 - In the **Select an Entity Property** list, select an entity property.
 - Click the Select Category Attribute search button. The Select a Category window pops up.
 - Select one or more categories and click **Select**.
 - Click **Select Category Attribute** to list the attributes of the selected categories.
 - Select a required attribute and click **Add** to define the mapping. You can define multiple mappings.
 - You can delete a mapping by clicking **Remove**.

Note: Be sure that the property mapping done is Content Server (in business object type) - do not overlap with the property mapping done in Process Experience Administration. If the same property mapping is done at both ends, data will be overwritten with the latest update.

Chapter 13

EIS connectors

Enterprise Information System (EIS) entity connectors provide access to information in other systems. When you add an entity connector to a project, you can create entities that represent the information available in that connector's target system. The structure of an EIS entity is determined by the target repository and you cannot modify it. That is, you cannot add, remove, or modify its structural building blocks. However, you can add and modify decorative building blocks just as you would for a native entity. Once decorated, end users can access an EIS entity via Process Experience, just as they would access native entities.

There are two types of EIS connectors.

- [Custom EIS connectors](#)
- [Predefined EIS connectors](#)

Custom EIS connectors

The EIS connector framework enables application developers to develop connectors to specific external systems by implementing the interfaces provided by the framework. To develop an EIS connector, you need a good working knowledge of the Java programming language. The EIS connector framework provides tools for building a connector to bring information from the enterprise's other information systems as entities into the entity modeling environment and use them in an application.

Building a custom EIS connector

The target external systems for a custom EIS connector might be enterprise applications (ERPs such as SAP) and repositories (for example, SuccessFactors or SharePoint). All of these systems hold information that can participate in an application either during building of the application or at run time during the execution.

The steps to build an EIS connector and use EIS entities in an application are summarized as follows:

1. In CWS, create a new EIS connector and provide the properties required to connect to the external system.

2. Define the entities and provide the implementation for the Create, Read, Update, Delete and List operations.
3. In System Resource Manager, manage EIS repository configurations by configuring the properties defined in the connector.
4. Import EIS entities through the EIS repository reference by selecting the appropriate connector and corresponding repository configuration.
5. Decorate the EIS entities with building blocks.

Defining the EIS connector

To create an EIS connector:

1. Create a workspace in CWS and create a project within it.
2. Right-click the project you created and select **New > EIS Connector**.
The EIS Connector document page opens with two tabs: General and Properties.
3. On the General tab, provide a name, display name, and description for the EIS connector.
4. Click **Save**.

If the name is valid, the **Generate API** option is enabled. When you click **Generate API**, two folders are generated in the corresponding project.

- The `crosscontext` folder contains the Java archive definition instance with the name of the EIS connector instance name appended with `_java`. This contains two jar dependencies:
 - `com.cordys.entity.services.eisconnector.jar`
 - `entityruntime-bean-interface.jar`.
- The other folder has the same name as the EIS connector. It contains a folder structure equal to the package structure used by the generated Java classes. It also contains the Definition Provider and Repository Provider Java classes.

Based on the name you provide, the EIS definition provider class, EIS repository provider class, and package structure are updated automatically.

For example, when the name of the EIS connector is 'Sample', the generated java classes are:

- EIS definition provider class:
`com.opentext.entity.eisconnector.sample.SampleEISDefinitionProvider`
- EIS repository provider class:
`com.opentext.entity.eisconnector.sample.SampleEISRepositoryProvider`

5. On the Properties tab, use the (Add) and (Remove) icons to add and remove properties.
These properties will be used to connect to the external system and can be provided with

a value by an administrator. Each property has a name, a display name, a type, and a default value. The name of the property is unique. The display name and default value are optional. You can add a language specific name for the display name by adding translations on the project.

There are three types of properties:

- String – This property type is used for a text value.
- Password - This property type is stored in Base64 encoded format. The primary purpose of the encoding is to protect the confidentiality of the property value stored in the XMLStore. Passwords are masked when provided a value.
- Number – This property type is used for an integer value.

Implementing the generated Java classes

After you generate the API, the next step is to synchronize the project to your file system and to implement the Java classes.

The generated source code will be available at the following location:

```
<Cordys_Installation_
Directory>\cws\sync\system\<CWSTWorkspace>\<Project>\<EISConnectorName>
```

To synchronize the project:

1. Create a Java project in IDE (for example, Eclipse) and link the generated source code to this Java project.
2. In the build path/class path, add `entityruntime-bean-interface.jar` available in the `crosscontext` folder of the Process Platform installation directory.

To implement methods in the generated Java classes:

- Implement all the methods in the two generated classes.
See [Implementing the Definition Provider class](#) and [Implementing the Repository Provider class](#) for details.

Implementing the Definition Provider class

The Definition Provider class contains the definitions of the entities and their properties.

The generated Definition Provider class looks as follows:

```
import
com.opentext.cordys.entityCore.connectors.genericEISConnector.definition.EISEntity;
import
com.opentext.cordys.entityCore.connectors.genericEISConnector.definition.IEISDefinitionProvider;
import java.lang.String;
```

```
import java.util.List;
import java.util.Map;

public class SampleEISDefinitionProvider implements IEISDefinitionProvider {
    @Override
    public void init(Map<String, String> map1)
    {
        // TODO Auto-generated method
    }
    @Override public EISEntity getEntity(String string1)
    {
        // TODO Auto-generated method
        return null;
    }
    @Override public List<EISEntity> getEntities() {
        // TODO Auto-generated method
        return null;
    }
}
```

Defining the entities

From the generated Java classes, identify the Definition Provider class and provide the implementation for the methods.

A sample implementation of the methods looks as follows:

```
import
com.opentext.cordys.entityCore.connectors.genericEISConnector.definition.EISEntity;
import
com.opentext.cordys.entityCore.connectors.genericEISConnector.definition.IEISDefinitionProvider;
import java.lang.String;
import java.util.List;
import java.util.Map;

public class SampleEISDefinitionProvider implements IEISDefinitionProvider
{
    @Override
    public void init(Map<String, String> map1)
    {
        // TODO Auto-generated method
    }
    @Override public EISEntity getEntity(String string1) {
        // TODO Auto-generated method
        return null;
    }
    @Override
    public List<EISEntity> getEntities() {
```

```
// TODO Auto-generated method  
return null; } }
```

In the above implementations, the `EISEntity` and `EISEntityProperty` classes are used to define the entity and its properties.

`EISEntity` has the following properties:

- `name` is the name of the EIS entity.
 - `localID` is used to identify the EIS entity while fetching its list. It should have a unique value.
 - `id` is a unique identifier for the EIS entity in the external system.
 - `label` is used to store the Display Name.

Note: It is mandatory to set the name and localId while defining an entity.

Each EIS entity must have at least one property and it is mandatory to define of the properties as a primary key using the `setPrimaryKey` method on the `EISEntityProperty` instance.

The `EISEntityProperty` instance defines a property of the EIS Entity. It defines a property with name, id, display name, data type, and max and min values for an integer type, length for a Text/Long Text type, precision for a float type, and default value and values list for enumerated types, constraints, updatability, and so forth.

Reading the values of the connection properties

The Definition Provider class provides the `init` method. This method is used to read the values of the connection properties that can be used to establish a connection to the external system.

A simple implementation of the `init` method follows.

```
/**  
 * @param map of key, value pair of property name and its values.  
 */  
public void init(Map<String, String> propertiesMap)  
{  
    this.userName = propertiesMap.get("userName");  
    this.password = propertiesMap.get("password");  
}
```

Note: Even though properties of type Password are stored encoded in the XMLStore, the propertiesMap contains the decoded values.

Retrieving EIS entity definitions

The `getEntities` method in the Definition Provider class is used by the system to retrieve all EIS entities definitions as a list of `EISentity` objects.

A sample implementation of the `getEntities` method follows.

```
@Override  
public List<EISEntity> getEntities()  
{  
    List<EISEntity> entityList = new ArrayList<EISEntity>();  
    /* define entity */  
    SampleEISDefinitionProvider sampleEISDefProvider = new SampleEISDefinitionProvider();  
    /* add entity to list */  
    entityList.add(sampleEISDefProvider.build("Student"));  
    entityList.add(sampleEISDefProvider.build("Teacher"));  
    return entityList;  
}
```

The `getEntity` method in the Definition Provider class is used to retrieve the definition for a specific entity. It receives the entity name as input argument and returns the definition of the EIS entity as `EISEntity` object.

A sample implementation of the `getEntity` method follows.

```
@override  
public EISEntity getEntity(String string1)  
{  
    SampleEISDefinitionProvider sampleEISDefProvider = new SampleEISDefinitionProvider();  
    switch (string1)  
    {  
        case "Student":  
            return sampleEISDefProvider.build("Student");  
        case "Teacher":  
            return sampleEISDefProvider.build("Teacher");  
        default:  
            return new EISEntity();  
    }  
}
```

In the following sample, the `build` method is used as a utility method to construct the `EISEntity` objects. You can use this sample directly in the respective methods or write the required logic yourself.

A sample implementation of the `build` method follows.

```
EISEntity build(String entityName)  
{  
    EISEntity eisEntity = new EISEntity();  
    eisEntity.setName(entityName);  
    // create a property 'id'  
    EISEntityProperty id = new EISEntityProperty();
```

```

id.setId("id");
id.setName("id");
id.setChangeability(Changeability.ONLY_WHEN_FIRST_CREATED);
id.setType(Type.INTEGER);
id.setDescription("id");
id.setPrimaryKey(true);
// create a property 'name'
EISEntityProperty name = new EISEntityProperty();
name.setId("name");
name.setName("name");
name.setChangeability(Changeability.ANY_TIME);
name.setType(Type.TEXT);
name.setLength(60);
name.setDescription("name");
name.setPrimaryKey(false);
// add property to entity
eisEntity.addProperty(id);
eisEntity.addProperty(name);
return eisEntity;
}
}

```

Sample to create an entity

The following sample shows how you can create an `EISEntity` object.

```

EISEntity eisEntity = new EISEntity();
eisEntity.setName("Employee");

```

Static and dynamic entities

The EIS connector framework supports both static and dynamic entities.

- Static entity - The EIS connector has knowledge of the EIS entity even without connecting to the external system (offline information of the entity definition is stored in the connector itself). For example, an ERP EIS connector having an Invoice entity.
- Dynamic Entity - The EIS connector has no knowledge of the EIS entity to be created without connecting to the external system (no offline information is possible until it is connected to the external system and it does not know the information about the EIS entities).

For example, a database EIS connector can recognize each table as an EIS entity. This connector needs to connect to the external system in order to get the entity (table) definitions.

The EIS connector framework does not differentiate between static and dynamic entities, as `getEntities` just returns the list of entities defined by this connector.

If the `getEntities` method can utilize connection information initialized in the `init` method, it can connect to the external system and build the entity definition dynamically. In this way, dynamic entities can be defined.

Properties

The following sections provide samples, notes, and Java types for creating various types of properties.

Boolean

Sample

```
EISEntityProperty isFullTimeEmployee = new EISEntityProperty();
isFullTimeEmployee.setId("isFullTimeEmployee");
isFullTimeEmployee.setName("isFullTimeEmployee");
isFullTimeEmployee.setChangeability(Changeability.ANY_TIME);
isFullTimeEmployee.setType(Type.BOOLEAN);
isFullTimeEmployee.setDescription("isFullTimeEmployee");
isFullTimeEmployee.setPrimaryKey(false);
isFullTimeEmployee.setFalseDisplayName("False");
isFullTimeEmployee.setNoneDisplayName("None");
isFullTimeEmployee.setTrueDisplayName("True");
isFullTimeEmployee.setFalseDisplayName("False");
isFullTimeEmployee.setNoneDisplayName("None");

isFullTimeEmployee.setTrueDisplayName("True");
```

Notes

`setChangeability` accepts the following values:

- ANY_TIME - The property value can be updated at any time.
- NEVER - Use this for ReadOnly properties.
- ONLY_WHEN_FIRST_CREATED - The value can be set for this property only on its creation.

`setNoneDisplayName` adds display name None to the none value.

`setFalseDisplayName` adds display name False to the false value.

`setTrueDisplayName` adds display name True to the true value.

Java type

`java.lang.Boolean`

Currency

Sample

```
EISEntityProperty salary= new EISEntityProperty();
salary.setId("salary");
salary.setName("salary");
salary.setChangeability(Changeability.ANY_TIME);
salary.setType(Type.CURRENCY);
```

```

Salary.setCurrencyCode("INR") salary.setLength(10);
salary.setScale(5);
salary.setMaximum("100");
salary.setMinimum("0");
salary.setDescription ("salary");

salary.setPrimaryKey(false);

```

Notes

`setLength` sets the total number of digits of decimal.

`setScale` specifies the scaling.

`setMaximum` and `setMinimum` set the minimum and maximum range of the number.

`setCurrencyCode` sets the ISO code of the corresponding currency.

See [Adding properties](#) for more information about the Currency data type.

Java type

`java.math.BigDecimal`

Date

Sample

```

EISEntityProperty dob = new EISEntityProperty();
dob.setId("dob");
dob.setName("dob");
dob.setChangeability(Changeability.ANY_TIME);
dob.setType(Type.DATE);
dob.setDescription("dob");

dob.setPrimaryKey(false);

```

Java Type

`java.util.Date`

Date_time

Sample

```

EISEntityProperty dob = new EISEntityProperty();
dob.setId("dob");
dob.setName("dob");
dob.setChangeability(Changeability.ANY_TIME);
dob.setType(Type.DATE_TIME);
dob.setDescription("dob");

dob.setPrimaryKey(false);

```

Java type

java.util.Date

Decimal**Sample**

```
EISEntityProperty bonus = new EISEntityProperty();
bonus.setId("bonus");
bonus.setName("bonus");
bonus.setChangeability(Changeability.ANY_TIME);
bonus.setType(Type.DECIMAL);
bonus.setLength(10);
bonus.setScale(5);
bonus.setMaximum("100");
bonus.setMinimum("0");
bonus.setDescription("bonus");

bonus.setPrimaryKey(false);
```

Notes

`setLength` is used to set the total number of digits of decimal.

`setScale` specifies the scaling.

By using `setMaximum` and `setMinimum`, you can set the minimum and maximum range of the number.

Java type

java.math.BigDecimal

Enumerated_integer**Sample**

```
EISEntityProperty noOfClassesPerDay = new EISEntityProperty();
noOfClassesPerDay.setId("noOfClassesPerDay");
noOfClassesPerDay.setName("noOfClassesPerDay");
noOfClassesPerDay.setChangeability(Changeability.ANY_TIME);
noOfClassesPerDay.setType(Type.ENUMERATED_INTEGER);
noOfClassesPerDay.setDescription("noOfClassesPerDay");
noOfClassesPerDay.setPrimaryKey(false);
noOfClassesPerDay.setLength(20);
ValueList noOfActivities = new ValueList();
List<EnumerationProperty> noOfActivitieslist = noOfActivities.getValue2();
for(int i=1;i<=5;i++)
{
    noOfActivitieslist.add(new EnumerationProperty(String.valueOf(i), "Class" +
String.valueOf(i)));
}
```

```
noOfClassesPerDay.setValueList(noOfActivities); noOfClassesPerDay.setDefaultValue("1");
```

Notes

Create a ValueList object. Create a list of EnumerationProperty with value and displayname. Add ValueList object to EISEntityProperty object.

By using setDefaultValue, you can set the default value among the valueList.

```
new EnumerationProperty(String.valueOf(i), "Class" + String.valueOf(i)));
```

Adds display name Class1 for EnumeratedProperty 1.

Java type

java.lang.Integer

Enumerated_text

Sample

```
EISEntityProperty gender = new EISEntityProperty();
gender.setId("gender");
gender.setName("gender");
gender.setChangeability(Changeability.ANY_TIME);
gender.setType(Type ENUMERATED_TEXT);
gender.setDescription("gender");
gender.setLength(15);
gender.setPrimaryKey(false);
ValueList genderList = new ValueList();
List<EnumerationProperty> list = genderList.getValue2();
list.add(new EnumerationProperty("Male", "Male_DisplayName"));
list.add(new EnumerationProperty("Female", "Female_DisplayName"));
gender.setValueList(genderList);

gender.setDefaultValue("Male");
```

Notes

Create a ValueList object. Create a list of EnumerationProperty with value and displayname. Add ValueList object to EISEntityProperty object. By using setDefaultValue, you can set the default value among the valueList.

```
list.add(new EnumerationProperty("Female", "Female_DisplayName"));
```

Adds displayname (Female_DisplayName) to the enumeration property (Female).

Java type

java.lang.String

Float**Sample**

```
EISEntityProperty remainingLeaves = new EISEntityProperty();
remainingLeaves.setId("remainingLeaves");
remainingLeaves.setName("remainingLeaves");
remainingLeaves.setChangeability(Changeability.ANY_TIME);
remainingLeaves.setType(Type.FLOAT);
remainingLeaves.setDescription("remainingLeaves");

remainingLeaves.setPrimaryKey(false);
```

Java type

java.lang.Float

Integer**Sample**

```
EISEntityProperty id = new EISEntityProperty();
id.setId("id");
id.setName("id");
id.setChangeability(Changeability.ONLY_WHEN_FIRST_CREATED);
id.setType(Type.INTEGER);
id.setDescription("id");

id.setPrimaryKey(true);
```

Notes

Set `setPrimaryKey` to true only if the property is a primary key.

Java type

java.lang.Integer

Long_text**Sample**

```
EISEntityProperty address = new EISEntityProperty();
address.setId("address");
address.setName("address");
address.setChangeability(Changeability.ANY_TIME);
address.setType(Type.LONG_TEXT);
address.setDescription("address");

address.setPrimaryKey(false);
```

Java type

java.lang.String

Text**Sample**

```
EISEntityProperty name = new EISEntityProperty();
name.setId("name");
name.setName("name");
name.setChangeability(Changeability.ANY_TIME);
name.setType(Type.TEXT);
name.setLength(60);
name.setDescription("name");

name.setPrimaryKey(false);
```

Notes

The Length field specifies the maximum length of the text property.

Java Type

java.lang.String

Implementing the Repository Provider class

The Repository Provider class contains the logic to:

- Connect to the external system
- Perform the CRUD operations
- Perform the list operation

The generated Repository Provider class looks as follows.

```
import com.opentext.cordys.entityCore.connectors.genericEISConnector.repository.*;
import java.util.List;

public class SampleEISRepositoryProvider implements IEISRepositoryProvider {
    @Override
    public void init(java.util.Map<java.lang.String, java.lang.String> map) {
        // Developer should implement this method for initializing the
        // connection by consuming the connection properties available in
        // the map defined in System Resource Manager.
    }
    @Override public boolean testConnection() {
        //TODO Auto-generated method return false;
    }
    @Override
```

```

        public EISEntityRow create(EISContext eiscontext, EISRepositoryEntity
eisrepositoryentity, EISEntityRow eisentityrow)
    {
        //TODO Auto-generated method return null;
    }
    @Override public EISEntityRow get(EISContext eiscontext, EISRepositoryEntity
eisrepositoryentity, EISEntityRow eisentityrow)
    {
        //TODO Auto-generated method
        return null;
    }
    @Override public boolean update(EISContext eiscontext, EISRepositoryEntity
eisrepositoryentity, EISEntityRow eisentityrow) {
        //TODO Auto-generated method
        return false;
    }
    @Override public boolean delete(EISContext eiscontext, EISRepositoryEntity
eisrepositoryentity, java.util.List<EISEntityRow> list) {
        //TODO Auto-generated method
        return false;
    }
    @Override public List<EISEntityRow> doQuery(EISContext eiscontext,
EISRepositoryEntity eisrepositoryentity, QueryWrapper querywrapper) {
        //TODO Auto-generated method
        return null;
    }
}
}
}

```

Note: Any runtime exception thrown will be displayed as an error to the end user who invoked the operation.

Implementing the test connection logic

In the Repository Provider class, the `testConnection` method must be implemented to check the connection with the external system. This method will be invoked by the Manage EIS Configurations screen in the System Resource Manager when clicking **Test Connection**.

A sample implementation of the `testConnection` method follows.

```

/** after init this is invoked to test the connection
 * @return true(connection success)/false(connection failed)
 */
@Override
public boolean testConnection() {
    // Implement the logic for connecting external system using init //properties.
    sample code is establishing connection to the database.
    try {

```

```

        connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1
this.userName, this.password);
        ...
    }
    catch (SQLException ex)
    {
        System.out.println("Connection Failed! Check output console");
        ex.printStackTrace();
    }
    if (connection != null) {
        return true;
    }
    else {
        return false;
    }
}

```

Reading the values of the connection properties

The Repository Provider class provides the `init` method, which can be used to read the values of the connection properties to establish a connection to the external system.

See [Reading the values of the connection properties in Implementing the Definition Provider class](#).

A sample implementation of the `init` method follows.

```

public void init(Map<String, String> map)
{
    this.userName = map.get("userName");
    this.password = map.get("password");
    ...
}

```

Implementing the CRUD operations

The following sections describe how to implement the Create, Read, Update, and Delete operations.

Create

To create an instance for an EIS entity, implement the `create` method in the Repository Provider class. This method will be invoked from the entity runtime when a user creates an instance of the entity (an item). The user needs to provide the required properties in the Create form in Process Experience.

```

public EISEntityRow create(EISContext eisContext1, EISRepositoryEntity
eisRepositoryEntity2, EISEntityRow eisEntityRow3)

```

Input parameters

- `eisContext1` - Contains details about the organization and the user who is trying to create the item.
- `eisRepositoryEntity2` - Provides details about the EIS entity, including `name`, `localId` (unique ID to represent the EIS entity in the external system), and so forth.
- `eisEntityRow` - Contains the input data provided by the user to create a new item.

Output parameter

- `EISEntityRow` - If the instance is successfully created, the `create` method returns all new instance details, including the `localId` that will be used to identify the entity in CWS.

EISContext

The `EISContext` class holds the current login information for the user who requested the operation. Using this context information, the EIS connector can propagate the user authentication to the target system.

`EISContext` provides the following public methods to retrieve the context information:

- `getOrganizationName` – Returns the Process Platform organization name
- `getOrganizationId` – Returns the Process Platform organization internal Id
- `getUserID` – Returns the Process Platform user Id
- `getPrincipal` – Returns a `javax.security.Principal` object received from the web server for the logged in user

Sample implementation for create

```
@Override
public EISEntityRow create(EISContext eisContext1, EISRepositoryEntity
eisRepositoryEntity2, EISEntityRow eisEntityRow3)
{
    try
    {
        switch (eisRepositoryEntity2.getLocalId())
        {
            case InvoiceEntity.INVOICE: /* Logic to call external system with all r
to create INVOICE entity/element*/
                break;
            case EmployeeEntity.EMPLOYEE: /* Logic to call external system with all
data to create EMPLOYEE entity/element*/
                break;
            default:
                throw new RuntimeException("Unknown entity type");
                break;
        }
        return eisEntityRow;
    }
    catch (Exception ex)
    {
```

```
        throw new RuntimeException(ex);
    }
}
```

Get

To read the details of an item, implement the `get` method in the Repository Provider class. This method is invoked when a user opens an item.

```
public EISEntityRow get(EISContext eisContext1, EISRepositoryEntity  
eisRespositoryEntity2, EISEntityRow eisEntityRow3)
```

Input parameters

- `eisContext` - Contains details about the organization and the user who is trying to view information about an item.
 - `eisRepositoryEntity` - Provides details about the EIS entity, including name, localId (unique id to represent the EIS entity in the external system), and so forth.
 - `eisEntityRow` - Contains the input data provided by the user to get an instance of the EIS entity.

Output parameter

- `EISEntityRow` - Returns an instance of `EISEntityRow` that contains the property details filled by fetching details from the external system.

Sample implementation for get

```
@Override
public EISEntityRow get(EISContext eisContext1, EISRepositoryEntity
eisRepositoryEntity2, EISEntityRow eisEntityRow3)
{
    try
    {
        EISEntityRow eisRetrievedRow = null;
        switch (eisRepositoryEntity2.getLocalId())
        {
            case InvoiceEntity.INVOICE:
                // Call External system API by passing details of entity detail
instance.
                break;
            case EmployeeEntity.EMPLOYEE: // Call External system API by passing de
entity details of EMPLOYEE instance.
                break;
            default:
                throw new RuntimeException("Unknown entity type");
                break;
        }
    }
```

```
        return eisRetrievedRow;
    }
    catch (Exception ex)
    {
        throw new RuntimeException(ex);
    }
}
```

Sample to create an `EISEntityRow` object

In the EIS connector implementation, the `EISEntityRow` object is built with the EIS entity data fetched from the external system. The `EISEntityRow` object represents an EIS entity instance, and is passed to the entity run time to be displayed in a form or as a row in result list in Process Experience.

As each entity will have a set of properties, an object is needed to hold the property details of an EIS entity. After fetching data of the EIS entity instance from the external system, first build an `EISEntityRowColumn` object for each property of the EIS entity. The property name will be held in the `columnName` field of `EISEntityRowColumn` and the property value will be held in the `columnValue` field of `EISEntityRowColumn`.

Then add all `EISEntityRowColumn` objects to a list and add that list to the `EISEntityRow` object.

```
public static Object getColumnValue(ResultSet rs, String column,
EISEntityProperty.Type dataType) throws SQLException
{
    switch (dataType) {
        case INTEGER:
            return rs.getInt(column);
        case TEXT:
            return rs.getString(column);
        case DECIMAL:
            return rs.getBigDecimal(column);
        case DATE:
        case DATE_TIME:
            java.sql.Timestamp timestamp = rs.getTimestamp(column, Calendar.getInstance(TimeZone.getTimeZone("UTC")));
            return timestamp;
        case ENUMERATED_TEXT:
            return rs.getString(column);
        case BOOLEAN:
            return rs.getBoolean(column);
        case ENUMERATED_INTEGER:
            return rs.getInt(column);
        case FLOAT:
            return rs.getFloat(column);
        case LONG_TEXT:
            return rs.getString(column);
        default:
            return null;
    }
}
```

Update

To update an instance of an EIS entity, provide implementation for the following Java method in the Repository Provider class. This method will be invoked when a user makes changes to an item.

```
public boolean update(EISContext eisContext1, EISRepositoryEntity
eisRespositoryEntity2, EISEntityRow eisEntityRow3)
```

Input parameters

- `eisContext1` - Contains details about the organization and the user who is trying to create an instance.
- `eisRespositoryEntity` - Provides details about the EIS entity, including name, localId (unique ID to represent the EIS entity in the external system), and so forth.
- `eisEntityRow` - Contains the input data provided by the user to update the instance of the EIS entity.

Output parameter

- boolean value - If the instance is successfully updated, this method returns true; if not, it returns false.

Sample implementation for update

```
@Override
public boolean update(EISContext eisContext1, EISRepositoryEntity
eisRespositoryEntity2, EISEntityRow eisEntityRow3)
{
    try
    {
        switch (eisRespositoryEntity2.getLocalId())
        {
            case InvoiceEntity.INVOICE:
                // Logic to call external system API to update the INVOICE EIS
                return true/false;
            case EmployeeEntity.EMPLOYEE: // Logic to call external system API to u
INVOICE EIS entity
                return true/false;
            default:
                throw new RuntimeException("Unknown entity type");
                break;
        }
        return true;
    }
    catch (Exception ex) {
        throw new RuntimeException(ex);
    }
}
```

Delete

To delete an instance of an EIS entity, implement the `delete` Java method defined in the Repository Provider class. This method will be invoked when a user removes an item.

```
public boolean delete(EISContext eisContext1,  
EISRepositoryEntity eisRespositoryEntity2,  
List<EISEntityRow> list3)
```

Input parameters

- `eisContext` - Contains details about the organization and the user who is trying to delete one or more instances.
- `eisRespositoryEntity` - Provides details about the EIS entity, including name, localId (unique ID representing the EIS entity in the external system), and so forth.
- `listEISEntityRows` - A list of instances in the form of `EISEntityRow` objects that contains the input data provided by the user to delete instances of the EIS entity.

Output parameter

- boolean value - If the instance is successfully deleted, this method returns true; if not, it returns false.

Sample implementation for delete

```
@Override public boolean delete(EISContext eisContext1, EISRepositoryEntity  
eisRespositoryEntity2, List<EISEntityRow> listEntityRows3)  
{  
    try  
    {  
        for (EISEntityRow eisEntityRow : listEntityRows3)  
        {  
            switch (eisRespositoryEntity2.getLocalId())  
            {  
                case InvoiceEntity.INVOICE: /* * Logic to delete instances of I:  
passed all * details of instance including Id which will be used to * identify the  
instance */  
                    return true/false;  
                case EmployeeEntity.EMPLOYEE: /* *Logic to delete instances of E:  
passed all *details of instance including Id which will be used to *identify the  
instance */  
                    return true;  
                default:  
                    throw new RuntimeException("Unknown entity type");  
                    break;  
            }  
        }  
    }  
    catch (Exception ex) {  
        throw new RuntimeException(ex);  
    }  
}
```

```

        return false;
}

```

Implementing the List operation

To display a list of EIS entity instances in the Results panel, implement the `doQuery` method in the Repository Provider class. The user can apply filters, sort on selected properties, and provide the pagination details in the Results panel. The external system may not support filters and sorting on all types of data, so you will need to handle that in the implementation.

```

public List<EISEntityRow> doQuery(EISContext eisContext1,
EISRepositoryEntity eisRepositoryEntity2, QueryWrapper queryWrapper3)

```

Input parameters

- `eisContext` - Contains details about the organization and the user who is trying to display the list of instances.
- `eisRespositoryEntity` - Provides details about the the EIS entity, including name, `localId` (unique ID representing the EIS entity in the external system), and so forth.
- `query` - Contains the filter, sorting details applied on selected entity properties, and pagination details to get the exact number of records to return from the `doQuery` method.

Output parameter

- `List<EISEntityRow>`- The list of items of the specified EIS entity.

The method `QueryWrapper.getJsonQuery` returns the JSON string of the list page.

The following are the main elements in the JSON query string from which the connector can prepare the results based on the query

- `select`
- `orderBy`
- `criteria`
- `skip`
- `top`

Sample JSON

```
{
  "parameters": {

```

```

},
"select": [
{
  "itemMemberId": "3417EBB0751F11E7E1177C8DDB56BF51",
  "propertyId": "3417EBB0751F11E7E1177C8DDB573F51",
  "name": "Properties.id"
},
{
  "itemMemberId": "3417EBB0751F11E7E1177C8DDB56BF51",
  "propertyId": "3417EBB0751F11E7E1177C8DDB57BF51",
  "name": "Properties.description"
},
{
  "itemMemberId": "3417EBB0751F11E7E1177C8DDB56BF51",
  "propertyId": "3417EBB0751F11E7E1177C8DDB5AFF51",
  "name": "Properties.etype"
},
{
  "itemMemberId": "3417EBB0751F11E7E1177C8DDB56BF51",
  "propertyId": "3417EBB0751F11E7E1177C8DDB593F51",
  "name": "Properties.gender"
},
{
  "itemMemberId": "3417EBB0751F11E7E1177C8DDB56BF51",
  "propertyId": "3417EBB0751F11E7E1177C8DDB5CFF51",
  "name": "Properties.rating"
},
{
  "itemMemberId": "3417EBB0751F11E7E1177C8DDB56BF51",
  "propertyId": "3417EBB0751F11E7E1177C8DDB58BF51",
  "name": "Properties.salary"
},
{
  "itemMemberId": "3417EBB0751F11E7E1177C8DDB56BF51",
  "propertyId": "3417EBB0751F11E7E1177C8DDB59BF51",
  "name": "Properties.bonus"
},
{
  "itemMemberId": "d85ca4de859b3c9fae9713997fe11ba2",
  "propertyId": "ED3AC7849F3A40B4ACFB111FE03592B8",
  "name": "Identity.OpenUri"
}
],
"orderBy": [
{
  "itemMemberId": "3417EBB0751F11E7E1177C8DDB56BF51",
  "propertyId": "3417EBB0751F11E7E1177C8DDB573F51",
  "name": "Properties.id",
  "sortType": "ASC"
}
]

```

```
{
  "type": "PropertyTerm",
  "id": "D6D18AC8054F4D85B0D521B9D0C6760C",
  "itemMemberPropertyId": {
    "itemMemberPath": [
      ],
      "itemMemberId": "3417EBB0751F11E7E1177C8DDB56BF51",
      "propertyId": "3417EBB0751F11E7E1177C8DDB573F51"
    },
    "itemMemberId": "3417EBB0751F11E7E1177C8DDB56BF51",
    "propertyId": "3417EBB0751F11E7E1177C8DDB573F51",
    "name": "Properties.id",
    "comparison": {
      "operator": "Between",
      "from": 1,
      "to": 10,
      "isFromExpression": false,
      "isToExpression": false
    }
  },
  "skip": 0,
  "top": 2,
  "resultOption": "Items",
  "SearchAllVersions": false,
  "distinct": false
}
```

select

The select element is an array element that has the following structure.

```
"itemMemberId": "xxxxxxxxxxxx",
"propertyId": "xxxxxxxxxxxx",
"name": "Properties.id"
```

Note: The name property provides the external property name of the EIS entity. It starts with the Properties keyword, as these come from the Properties building block.

orderBy

The orderBy element is an array element that has the following structure.

```
"itemMemberId": "xxxxxxxxxxxx",
"propertyId": "xxxxxxxxxxxx",
"name": "Properties.id"
"sortType": "ASC"
```

Note: The `name` property provides the external property name of the EIS entity. It starts with `Properties`, as these come from the Properties building block. The `sortType` specifies the sorting order of the property. It can have the following values:

- `ASC` - Ascending order
- `DESC` - Descending order

criteria

`criteria` is an optional element that can be empty. `criteria` has a tree structure that can contain `PropertyTerm` or `LogicalOpTerm` elements that are referred to as `terms`.

PropertyTerm

`PropertyTerm` is a leaf node so it cannot contain any other term nodes.

```
"type": "PropertyTerm",
... "name": "Properties.id",
"comparison": {
  "operator": "Between",
  "from": 1,
  "to": 10,
  "isFromExpression": false,
  "isToExpression": false
}
```

The `name` property provides the external property name of the EIS entity. It starts with `Properties`, as these come from the Properties building block.

`PropertyTerm` has a comparison node.

Comparator	Description	Sample SQL	
NotInList	Not in (x1, x2, x3....)	x not in (1, 2)	<pre>"comparison": { "operator": "NotInList", "values": [1, 2], "isValueExpression": false }</pre>
Like	Like	x like '%nar%'	<pre>"comparison": { "operator": "Like", "value": "nar", "isValueExpression": false }</pre>
IsNull	Is Null	x is null	<pre>"comparison": { "operator": "IsNull" }</pre>

Comparator	Description	Sample SQL	
			}
IsNotNull	Is Not Null	x is not null	"comparison": { "operator": "IsNotNull" }
InList	In (x1, x2, x3 ...)	x in (1, 2)	"comparison": { "operator": "InList", "values": [1, 2], "isValueExpression": false }
Between	Between x1 and x2	x between 1 and 10	"comparison": { "operator": "Between", "from": 1, "to": 10, "isFromExpression": false, "isToExpression": false }
ge	greaterthan or equal >=	x >= 1500	"comparison": { "operator": "ge", "value": 1500, "isValueExpression": false }
gt	greater than >	x > 1500	"comparison": { "operator": "gt", "value": 1500, "isValueExpression": false }
eq	equals =	x = 'FULLTIME'	"comparison": { "operator": "eq", "value": "FULLTIME", "isValueExpression": false }
le	lessthan or equal <=	x <= 1500	"comparison": { "operator": "le", "value": 1500, "isValueExpression": false }

Comparator	Description	Sample SQL	
lt	less than <	x < 1500	<pre>"comparison": { "operator": "lt", "value": 1500, "isValueExpression": false }</pre>
ne	not equals !=	x != false	<pre>"comparison": { "operator": "ne", "value": true, "isValueExpression": false }</pre>

LogicalOpTerm

LogicalOpTerm (AND, OR) can have other term nodes in the terms elements.

type can be either AND or OR.

terms is an array of other term nodes.

The type operator should be applied between all the terms for that node.

```

"type": "AND",
"id": "986A48DDCE294567B5C96D94AB166C6B",
"terms": [
  {
    ...
  }
] {
  "type": "AND",
  "terms": [
    {
      "type": "PropertyTerm",
      "name": "Properties.x1",
      "comparison": {
        "operator": "eq",
        "value": true
      }
    },
    {
      "type": "OR",
      "terms": [
        {
          "type": "PropertyTerm",
          "name": "Properties.x2",
        }
      ]
    }
  ]
}

```

```
"comparison": {
    "operator": "ge",
    "value": 100
}
},
{
    "type": "PropertyTerm",
    "name": "Properties.x3",
    "comparison": {
        "operator": "InList",
        "values": [
            1,
            2,
            3
        ]
    }
},
{
    "type": "AND",
    "terms": [
        {
            "type": "PropertyTerm",
            "name": "Properties.x4",
            "comparison": {
                "operator": "eq",
                "value": 1
            }
        }
    ]
},
{
    "type": "AND",
    "terms": [
        {
            "type": "PropertyTerm",
            "name": "Properties.x5",
            "comparison": {
                "operator": "eq",
                "value": 2
            }
        },
        {
            "type": "PropertyTerm",
            "name": "Properties.x6",
            "comparison": {
                "operator": "IsNull"
            }
        },
        {
            "type": "PropertyTerm",
            "name": "Properties.x7",
            "comparison": {
                "operator": "eq",
                "value": 1
            }
        }
    ]
}
```

```
    }  
}  
]  
}  
}  
}  
}
```

The following example is the equivalent of the SQL query for the above JSON query criteria object. It is provided for better understanding of the JSON query.

```
(x1 = true) AND ( (x2 >= 100) OR ( x3 IN (1, 2, 3) OR ( x4 = 1) OR ( ( x5 =2) AND (x6 IS NULL) OR ( x7 NOT INT ('Fred', 'Ralph') ) )
```

The following table explains how to convert from the JSON string to the corresponding EntityProperty type value.

EISPropertyType	Converter code
Boolean	Boolean.parseBoolean
Date	Calendar date = DatatypeConverter.parseDateTime(s); date.setTimeZone(TimeZone.getTimeZone("UTC")); date.getTime();
DateTime	Calendar date = javax.xml.bind.DatatypeConverter.DatatypeConverter.parseDateTime (s); date.setTimeZone(TimeZone.getTimeZone("UTC")); date.getTime();
Decimal	javax.xml.bind.DatatypeConverter.DatatypeConverter.parseDouble
EnumeratedInteger	Integer.parseInt
EnumeratedText	String
Float	Float.parseFloat
Integer	Integer.parseInt
LongText	String
Text	String

Skip

Skip is used in conjunction with Top to implement the pagination behavior. `skip` is an integer value, indicating the number of records that need to be skipped.

Top

Top is used in conjunction with Skip to implement the pagination behavior. `top` is an integer value, indicating that this many records need to be returned (after the `skip`).

Example

For example, the following combination of `skip` and `top` returns records from 200 to 220.

```
skip 200
top 20
```

Note: If `top` is 0, it returns all the results.

Exception handling

In the Repository Provider class, implementation methods need to throw a `RuntimeException` object in the following cases.

- When not able to find the requested entity instance in the `get` method.
- When not able to create or update entity instances of the requested EIS entity.
- In the `doQuery` method, just return an empty list if instances for requested criteria are not found.

Packaging the application

After synchronizing the classes to CWS, the EIS connector is ready to be packaged.

To synchronize the project back to the CWS development workspace:

1. Copy all the required external jars to the `crosscontext` folder of the CWS project.
2. Add these jars as dependencies to the Java Archive Definition in the CWS project.

To package the application:

- Right-click the CWS project and then go to **Packaging > Create Package**.

See *Creating and Downloading Application Packages* in the Process Platform product documentation.

Using EIS entities in an application

You need to perform certain tasks to use EIS entities in an application.

Deploying the EIS connector package

To deploy applications using Application Deployer:

- See *Deploying Applications using Application Deployer* in the Process Platform product documentation.

After you deploy the package:

1. Go to System Resource Manager.
2. Right-click the Collaborative Workspace soap processor and select **Properties**.
3. On the JRE Configuration tab, update the classpath with <CORDYS_HOME>/crosscontext/*.jar.
4. Replace <CORDYS_HOME> with the corresponding Process Platform Installation directory.
5. Restart the Collaborative Workspace service container.
6. Restart TomEE.

Configuring the connection

To configure the connection to the external system, a new EIS repository configuration must be created.

You can create, update, or delete an EIS repository configuration.

To create an EIS repository configuration:

1. In System Resource Manager, click  (Manage EIS Repository Configurations).
2. Click  (Add).
3. Type a name and description for the configuration.
4. Select a connector type.
The properties that are displayed are based on the selected connector.
5. Provide the values for the properties.
6. Click  (Test) to test the configuration.
7. Click  (Save).

To update an EIS repository configuration:

1. In System Resource Manager, click  (Manage EIS Repository Configurations).
2. Select a row from the grid.
The details of the repository configuration are displayed below the grid.
3. Make the necessary changes.
4. Click  (Test).
5. Click  (Save).

To delete an EIS repository configuration:

1. In System Resource Manager, click  (Manage EIS Repository Configurations).
2. Select the check boxes of the rows to delete and click  (Delete).
3. When prompted for confirmation, click **Yes** to delete the selected rows or **No** to cancel the deletion.

Creating an EIS repository reference and importing entities

The EIS repository reference is responsible for importing EIS entities into the CWS project folder. It needs the EIS connector and its corresponding EIS repository configuration to connect to the external system.

To create an EIS repository reference and import entities:

1. In CWS, right-click the project folder, click **New > EIS Repository Reference**.
2. On the General tab, provide the display name, name, and description of the EIS repository reference.
3. Click **Import Entities**.
4. In the Import External Entities window, select an EIS connector and its corresponding repository configuration that was created in System Resource Manager.
5. Click  (Save).
6. Click **Load entities**.
The EIS entities are listed on the right side of the page.
7. In the list of loaded entities, search and select the check boxes for the entities to import into the CWS project.
8. Click **Synchronize**.
A folder with the same name as the EIS repository reference is created in the project and all imported EIS entities are placed inside it.

Building the application

The structure of an EIS entity is determined by the EIS connector implementation and you cannot modify it from CWS. That is, you cannot add, remove, or modify its structural building blocks. However, you can add and modify decorative building blocks just as you would for a native entity. Once decorated, users can access an EIS entity via Process Experience, just as they would access native entities.

To build the application:

1. Create the EIS repository reference and import the EIS entities.
See [Creating an EIS repository reference and importing entities](#).
2. Decorate the EIS entities by adding building blocks.
You cannot add properties or relationships to EIS entities. However, you can provide display names for the EIS entity properties.
3. Optionally, establish a relationship to an EIS entity or use EIS entities on the form of another entity.
4. Package the application.
The application package has a dependency on the EIS connector package, so it cannot be deployed without it.

Supported building blocks

The following building blocks can be added to EIS entities:

- [Action bar](#)
- [Form](#)
- [Layout](#)
- [List](#)
- [Security](#)

Relationships

To One and To Many relationships can have an EIS entity as target. An EIS entity itself cannot have outgoing relationships.

Deploying and maintaining the application

You need to perform certain tasks to deploy and maintain the application.

Deploying the application package

To deploy the application package:

- See *Deploying Applications using Application Deployer* in the Process Platform product documentation.

Maintaining the connection to the repository

Using the Process Experience Administration portal, an EIS repository reference can be associated with the appropriate EIS repository configuration after deployment.

To associate an EIS repository with the EIS repository configuration:

1. Open the Process Experience Administration tool. See [Administering an application](#).
2. Select **Global** or **Organization**, depending on the space where the package was deployed.
3. In the Workspaces pane, select the workspace with the deployed package name.
4. In the Configurable Elements pane, expand Repository References and click the repository reference.
5. On the Configuration tab, select an EIS connector and EIS repository configuration, and then click **Synchronize repository**.

Note: The Entities tab does not list any available entities until you click **Synchronize repository**.

6. On the Entities tab:
 - a. Click the project.
 - b. Provide **Use solution security** to the entities.
 - c. Click **Repository references** and select the repository reference.
 - d. Select the appropriate connector and repository configuration.
 - e. Click **Synchronize repository** to synchronize the loaded entities.

Changing the EIS connector

You may need to make changes to the EIS connector, such as removing properties or adding entities.

Note: This may break an existing application that uses the EIS connector.

To create a new version of the EIS Connector:

1. Open the existing EIS connector artifact and go to the Properties tab.
2. Add, change, or remove properties and click  (Save).
3. Update the implementation (Java sources) accordingly.
4. Right-click the project, select **Packaging > Package Properties**, change the version, and click **OK**.
5. Right click the project and select **Packaging > Create Package**.
6. Click **Download Package** to retrieve the package.
7. Deploy the new version of the EIS connector.

System Resource Manager changes

1. After deployment of the new version of the EIS connector, click  (Manage EIS Repository Configurations).
2. Identify the configurations for which the connector type is available.
3. Reselect (clear and then select) the connector type and then update the values of properties.
4. Click **Save**.

EIS repository reference project changes

1. After deployment of the new version of the EIS connector, open the EIS repository reference in the application project.
2. Click **Load entities**,
The entities are displayed in the right pane.
3. Select the entities and click **Synchronize**.
4. Click **Close**.
The EIS entities available in the project are updated.

5. Update the EIS entity's building blocks, if needed (with updated properties).
6. Right-click the project, and select **Packaging > Create Package**.
7. Click **Download Package** to retrieve the package.
Note: Synchronize creates the EIS entities if they are not available and also updates the existing EIS entities.
8. Deploy the new version of the EIS repository reference package.

Process Experience Administration

1. In the Process Experience Administration tool, select the workspace.
2. Expand the EIS repository reference.
3. Click the EIS repository reference instance.
4. Select the EIS connector and EIS repository configuration (if fields are empty).
5. Click **Synchronize**.

The list should appear in the run time with the updated definitions.

Predefined EIS connectors

Enterprise Information Service (EIS) entity connectors provide access to information in other systems. When you add an entity connector to a project, you can create entities that represent the information available in that connection's target system.

Note: Web services on EIS entities are not supported.

The functionality available to forms created for EIS entities aligns with the capabilities of the EIS connector that enables those entities. For instance, you may be able to create a form for an EIS entity such as those provided by the Case360, MBPM, PCL, or Inbox connections. However, if the connector does not allow create, update, or delete, the form allows only read access of the external data.

Some building blocks are created automatically when you create an entity connection. Although they are shown in the list of available building blocks, they have no designer configurable options. Therefore, they are not described as a separate topic in the Building Blocks section. For example, options called Create ToDo List and Create Watch List are available. If you select Create ToDo List, a building block called UserTask is automatically added to the list of building blocks. This building block is used to expose the list of properties you can use in a list or form that is created for the ToDoList and does not provide any options that you can configure.

Connections to the following internal systems are provided with Process Platform:

- Inbox
- Process Component Library

Process Platform also provides connections to the following external repositories:

- MBPM
- Case360

The first step in creating a customized user interface for an existing application is to create a connection to the external system.

[Creating a connection to an internal system](#)

[Creating an MBPM connection](#)

[Creating a Case360 connection](#)

Creating a connection to an internal system

You can create connections to the following internal systems:

- The Process Component Library entity connection enables access to data in a Process Component Library repository. When you create this type of connection, the system automatically creates ToDo List and Watch List entities that contain all of the properties from the external system.
- The Inbox entity connection provides access to tasks in the Inbox of Process Platform, thereby enabling entity-based applications to integrate with that part of the platform. When you create this type of connection, the system automatically creates a ToDo List entity that contains all of the properties from the external system.

Note: Properties from a ToDoList entity created from the InBox EIS connection do not support searching or sorting.

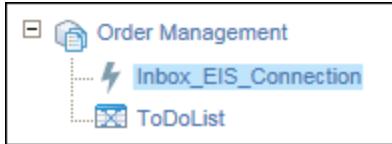
To create a connection to an internal system entity:

1. In **Workspace Documents**, select your project.
2. Click **New**.
3. Select **Other**, search for Process Platform EIS Connection. and select it from the results.
4. In Connection Properties, type the Display Name, Name, and Description of the connection.
5. In the Connection Type list, select **InBox** or **Process Component Library**.
6. Click the options corresponding to the entities you want to add to the project.
Clicking an option opens the entity modeler, where you can add building blocks to the ToDoList or WatchList entity.
 - For an Inbox connection, click **ToDo List**.
A UserTask building block is added to the ToDoList entity.
See the section in this topic called [Inbox properties](#) for a list of the properties that are exposed by this building block.
 - For a Process Component Library connection select either or both of the following options:
If you click **ToDo List**, a UserTask building block is added to the ToDoList entity.
If you click **WatchList**, a WorkItem building block is added to the WatchList entity.

See [Process Component Library properties](#) for a list of the properties that are exposed by these building blocks.

7. Save the changes to the entity, close the entity modeler, and close the Connection Properties.

The entity connection is shown in the project content tree in **Workspace Documents**. The entities that you chose to create are also listed in the project. For example, if you chose ToDo List, an entity called ToDoList is shown.



If you open a ToDoList or WatchList entity, you will see the automatically created UserTask and Workitem building blocks. These building blocks expose the properties that were defined in the external system. You cannot modify these properties but you can use them to create lists, forms, and layouts.

Inbox properties

For an Inbox connection, the UserTask building block exposes the following properties:

- TaskId
- Priority
- Status
- ActivityName
- ProcessName
- AssigneeDisplayName
- ReceivedDate
- StartDate
- DueDate

Process Component Library properties

For a Process Component Library connection, the UserTask and Workitem building blocks expose the following properties:

- Subject
- Reference
- DisplayReference
- Priority
- Deadline
- Message
- ServiceType

- Process
- CreatedOn
- CreatedBy
- LastUpdatedOn
- LastUpdatedBy
- AssignedTo
- Team
- AssigneeType
- LockedBy
- Status
- State
- Category
- Subcategory
- Field1
- Field2
- Field3
- Field4
- Field5

Creating an MBPM connection

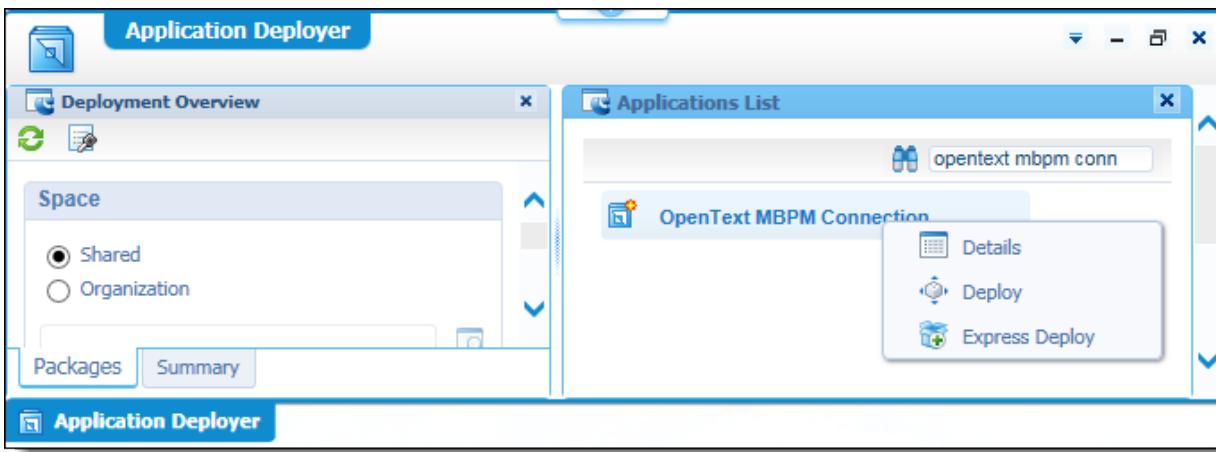
Use the MBPM Connection option to create entity types that model the information in the OpenText MBPM system. This enables users to retrieve and work with data in the MBPM system from Process Experience.

Note: The MBPM connection supports MBPM 9.4. Other versions of MBPM are not supported.

To create an EIS connection to MBPM, you must first deploy the cap files using Process Platform Application Deployer (if necessary).

To deploy the MBPM cap files:

1. Open Application Deployer.
2. In the list of connections, type OpenText MBPM Connection.
3. Right-click and select **Deploy** or **Express Deploy**.



4. In Step 1 of 5 through Step 3 of 5, click **Next**.
5. In Step 4 of 5, click **Deploy**.
6. When the process is complete, click **Finish**.
7. Close the Application Deployer.

To create an MBPM connection:

1. In the workspace, refresh the display.
2. Select your project and click **New**.
3. Select **Other**, search for MBPM EIS Connection, and select it from the results.
4. In Connection Properties, type the Display Name, Name, and Description of the connection.
The name can contain up to 128 characters. Valid characters are A-Z,a-z,0-9,_
5. Click (Save).
6. One by one, click each entity that you want to add to the project and click **Save**.
 - If you select ToDo List or Watch List, an MbpmAlert building block is automatically added. This option is used to expose the properties for MBPM Alert, which are used in the ToDo List and Watch List.
 - If you select Admin Forms, Blank Forms, or Reports, an MbpmForm building block is automatically added. This option is used to expose the properties for MBPM Form, which are used in AdminForms, BlankForms, and Reports.
7. Click (Save).
A summary of the information you entered is displayed.
8. If necessary, make any changes and then click **OK**.
The entity connection is added to the project content tree in **Workspace Documents** view. The entities that you chose to create are also listed in the project. For example, if you chose To Do List, an entity called ToDoList is shown. You can now start decorating the entities with lists, forms, and layouts.

Deleting an MBPM connection and its entities

To delete an MBPM connection and its entities:

Note: Before you delete an MBPM connection, you should delete all of its entities.

1. In the workspace used to create the connection, select the project.
2. Right-click each EIS entity and select **Delete** and then click **Yes** when prompted for confirmation.
3. Right-click the MBPM Connection and select **Delete**.

A message appears if entities are still associated with the connection. To see where the entities are used, click **Show Used by**.

When you are ready to delete the connection, click **Yes** to confirm.

Note: To also remove the MBPM Connection and its entities from Process Experience, you must delete the solution using the Process Experience [Administering an application](#).

Entity Types

When you create a connection that provides access to the information on an MBPM system, the following EIS entity types are automatically created in the entity run time where you create the MBPM connection.

Entity Type	Description
To Do and Watch Lists	Model the equivalent lists on the MBPM server. The items in the lists cannot be modified.
Blank Forms	Returns a list of MBPM Blank forms. A user creates new MBPM process instances by using the Blank Forms list. When a user clicks a form in the result list of the Results panel, the appropriate MBPM user interface is launched.
Administration Forms	Returns a list of MBPM Administration forms. When a user clicks a form in the result list of the Results panel, the appropriate MBPM user interface is launched.
Reports	Returns a list of MBPM reports. When a user clicks a report in the result list of the Results panel, the appropriate MBPM user interface is launched.

Properties

When you create a connection that provides access to the information on an external MBPM system, the following properties are automatically created in the entity run time where you create the MBPM connection.

To Do List and Watch List properties

The To Do List and Watch List EIS entities support the following properties. These properties are exposed by the MBPMAgent building block, which is automatically created and added to

the To Do and Watch List entities.

Property	Data Type	Length
AlertMessage	Text	250
Deadline	DateTime	
FolderId	Text	31
FolderName	Text	31
Priority	Int	
ProcessCaption	Text	250
StageCaption	Text	250
Subject	Text	250
Updated	DateTime	

Admin Forms, Blank Forms, and Reports properties

Admin Forms, Blank Forms, and Reports support the following properties. These properties are exposed by the MBPMForm building block, which is automatically created and added to the Blank Forms, Administration Forms, and Reports EIS entities.

Property	Data Type	Length
ActionCaption	Text	250
ActionName	Text	250
Description	Text	250
FormName	Text	31
GroupName	Text	31
ProcessCaption	Text	250
ProcessName	Text	250

Lists

You can create new lists for MBPM entity types (if you have permission). For example, you can create a custom To Do list to show only work at a specific state and associate it with the To Do List entity type.

Configuring the MBPM connection for Process Experience

You use the Process Experience Administration tool to configure an MBPM connection so that it will be available within Process Experience.

To configure an MBPM connection in Process Experience:

1. Start the Process Experience [Administering an application](#).
2. In the Solutions panel, select the solution that was created when the project was published or when the application package was deployed in Process Platform Explorer.
3. In the Configurable Elements panel, select **MBPM Connection**.
4. Select the Configuration Properties panel.
5. In the API URL section, type the URL to the MBPM API (the default is `http://localhost/BPMMobileService/`).
6. In the Web Client Uri, section type the URL to the MBPM API (the default is `http://localhost/Metastorm/`).
7. In the Authentication section, select **OTDS or Proxy Login**.
 - If you select OTDS, in the MBPM Resource Name, enter the OTDS resource defined for MBPM in Process Platform Security Administration (see [Authentication support](#)).
 - If you select Proxy Login, enter the User Name and Password.

Authentication support

The MBPM connection supports OTDS and Proxy Login authentication.

OTDS - If OTDS integration is used, the OTDS key needs to map the MBPM resource name. See [To configure an MBPM connection for OTDS authentication](#).

Proxy Login - If Proxy Login is used, the User Name and Password must be provided. See [To configure an MBPM connection for Proxy authentication](#).

Note: Be sure that the sap script `eProxyLogin_web.js` is uploaded.

The following table shows the authentication methods supported for various MBPM versions.

Version	OTDS	Proxy Login
MBPM 9.4	✓	
MBPM 9.4 Service Pack 1	✓	✓
MBPM 7.6 Service Pack 4		✓

To configure an MBPM connection for OTDS authentication:

1. In Process Platform Security Administration, select the OTDS Resources tab and, within that, the Other tab.
2. Create an OTDS Resource for MBPM with the following details.
 - Resource Name
 - Space - Shared
 - OTDS Server URL
 - OTDS Resource ID

Caution: Be sure that the **Allow this resource to impersonate its own users** impersonation setting for the Process Platform OTDS Resource is selected (checked) in the OTDS Server.

For more detailed information, see the following topics in the Process Platform product documentation.

- OTDS Resources and Trust
- Managing OTDS Platform Resources
- Managing OTDS Resources

To configure an MBPM connection for Proxy authentication:

- The MBPM username and password must be provided while configuring the EIS MBPM integration. With the provided User Name and Password, the standard MBPM user credentials validation will be taken place while impersonating the Process Platform User. Additionally, UserName must be an MBPM MetastormAdministrator role.
- The Process Platform username (the end user who signed in to Process Platform - called the impersonating user in this context) must be present in the MBPM eUser table (similarly to the way that SSO users are in the eUser table). The validation checks to determine whether this user is present in the eUser table.

Creating a Case360 connection

Use the Case360 Connection option to create entity types that model the information in the Case360 system. This enables users to use Process Experience to retrieve and work with data in the Case360 system.

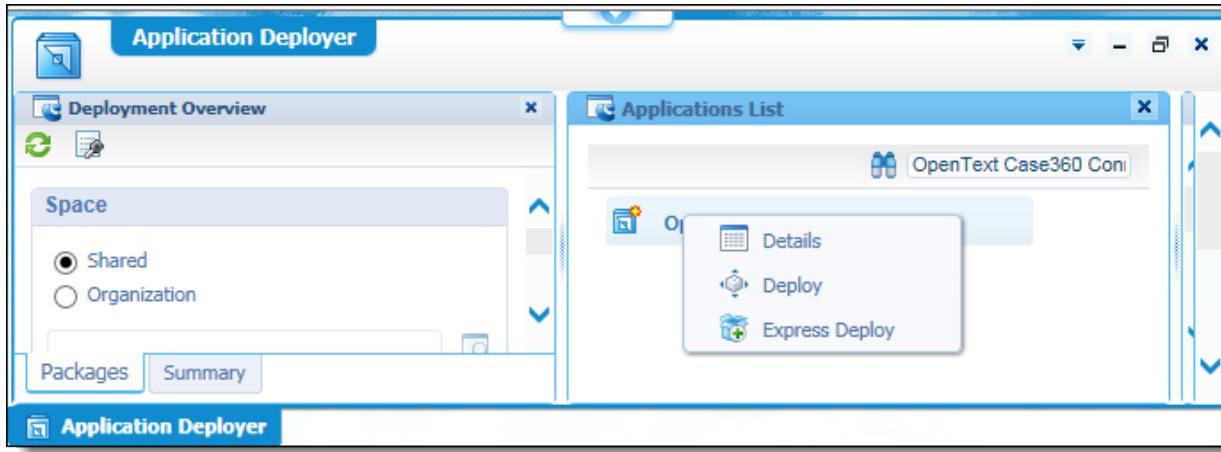
Note: The Case360 connection supports Case360 version 11.3.0.38307 or greater.

When you create a connection that provides access to the information on an external Case360 system, you can create EIS entities from any of the repository types listed in the Case360 Templates section.

To create an EIS connection to Case360, you must first deploy the cap files using Process Platform Application Deployer (if necessary).

To deploy the Case360 cap files:

1. Open Application Deployer.
2. In the list of connections, type OpenText Case360 Connection.
3. Right-click and select **Deploy** or **Express Deploy**.



4. In Step 1 of 5 through Step 3 of 5, click **Next**.
5. In Step 4 of 5, click **Deploy**.
6. When the process is complete, click **Finish**.
7. Close the Application Deployer.

To create a Case360 connection:

1. In **Workspace Documents**, refresh the display.
2. Select your project and click **New**.
3. Select **Other**, search for Case360 EIS Connection, and select it from the results.
4. In **Connection Properties**, enter the following information:
 - Display Name, Name, and Description. The name can contain up to 128 characters. Valid characters are A-Z,a-z,0-9,_
 - Server URL, Proxy ID, and Proxy Password of the Case360 system.

5. Click (Save).
6. Click (Refresh) next to Case360 templates.

Note: To get the list of Case360 templates, the ID used to log on to Process Platform must be defined in the Case360 system.

7. Click **Create** for each template for which you want to create an entity. Each EIS entity is created and displayed with the External Property building block that includes the list of its properties.
8. Click (Save).
A summary of the information you entered is displayed.
9. If necessary, make any changes and then click **OK**.
The entity connection is added to the project content tree in the **Workspace Documents** (Explorer) view. The entities that you chose to create are also listed in the project.

10. Click  (Quick Access Menu) > **Publish** to validate and enable the connection in Process Experience.

After an entity is created, a Sync option becomes available. Use this option to synchronize any changes to properties made in the Case360 templates after the connection was created.

To configure a Case360 Connection for Process Experience:

1. Start the Process Experience [Administering an application](#).
2. Select the solution and configurable elements.
3. In the Site URi box, type the URL to the Case360 system: <http://localhost:8080/sonora/>
4. Enter the ID and password used as the proxy ID to access the Case360 system.

To delete a Case360 connection and its EIS entities:

Note: Before you delete a Case360 connection, delete all of its entities.

1. In the workspace used to create the connection, select the project.
2. Right-click each entity and select **Delete** and then click **Yes** when prompted for confirmation.
3. Right-click the Case360 Connection and select **Delete**.
A message appears if entities are still associated with the connection. To see where the entities are used, click **Show Used by**. When you are ready to delete the connection, click **Yes** to confirm.

Note: To also remove the Case360 Connection and its EIS entities from Process Experience, you must delete the solution using the Process Experience [Administering an application](#).

Property mapping

Case360 Properties are mapped to the appropriate data type supported by Process Experience as follows:

Case360 Field Type	Process Experience Data Type
Boolean	Bool
Decimal	Decimal
Decimal Static Enumerated	Decimal
Decimal Dynamic Enumerated	Decimal
Duration	N/A
Integer Static Enumerated	Enumerated Integer
Integer Dynamic Enumerated	Enumerated Integer

Case360 Field Type	Process Experience Data Type
Repository Key	Text (Length 32)
Long Text	Long Text
XML	Long Text
Text	Text
Text Static Enumerated	Enumerated Text
Text Dynamic Enumerated	Text
TimeStamp Date	DateTime
TimeStamp DateTime	DateTime
TimeStamp Time Only	DateTime

Note: If a Boolean, Text Static Enumerated, or Integer Static Enumerated property is defined in the case template, the display names for the enumerated types must be entered for the property in the External Property building block for the EIS entity.

Lists

You can create new lists for Case360 EIS entity types (if you have permission). For example, you can create a custom list to show only work for the specific Case360 template using the custom properties from the repository type.

Note: Only custom properties for the selected template are supported (there is no support for Casefolder or process properties).

Chapter 14

Advanced functions

This section describes some advanced Entity Modeling functions.

Customizing the logo

You can customize the logo that appears in Process Experience in various ways. The following table provides a link to each method, shows what it changes, and shows the priority in which the logo customization is applied if multiple methods are used.

See the individual topic links in the Method column for complete instructions.

Method	What it changes	Priority
Adding a customized logo to a home page	Home page only.	1 - Takes precedence over any other supplied image
Creating a theme	The default logo for the selected organization.	2 - Individual home page layout logos take precedence over custom themes.

Defining a custom icon for a panel header

If the chrome for a panel is defined as Full, you can specify a custom icon for the panel header. Any other chrome options do not allow an icon to be selected.

Note: The image must come from the CWS project and the project must have a web library definition. See [Adding forms](#) for details.

If a custom panel icon is not defined, the default image is supplied for the Full chrome panel.

If a panel has a custom icon defined, changing the panel Chrome type removes the icon selection.

When a panel has an icon defined, the **Use default** option enables you to revert to the default icon.

In Process Experience, the icon is automatically sized to fit the space. To maintain picture quality, the recommended size for the icon is 42px x 42px.

Designing iHub reports on entity data

OpenText Process Suite supports OpenText Information Hub (iHub) and OpenText Analytics Designer.

- iHub is a scalable analytics and data visualization platform for designing, deploying, and managing secure interactive web applications, reports, and dashboards fed by multiple data sources.
- Analytics Studio, an integrated component of iHub, is a web-based modeler suited for the business analyst.
- Analytics Designer is an advanced desktop designer that enables developers to design and create dynamic reports with embedded analytics, data visualizations, interactive web 2.0 applications, and customizable dashboards for personalized insights and better end user experiences.

Using the iHub Analytics Studio or Analytics Designer, you can build reports based on entity data that is created when executing an entity-based application. Process Platform provides tools to simplify and speed up the process to build entity reports by generating a BIRT Data Object Design file that contains the database configuration(s) and references to the entity database tables. Entities in the solution are represented as data sets. Entity properties are represented as columns in data sets that you can use as measures and dimensions in a report. Relationships between entities are captured in the linked data model that is part of the same file.

Before you begin:

- Be sure that you have Administer Solution and Build permissions on the selected solution.

To export a Data Object Design file for a selected set of entities:

1. Open [Process Experience Administration](#) and select the solution under Workspaces.
2. Click  (Configuration).
3. Select **Export iHub data object design**.
4. Under **Select a fact table**, select the entity to use as a fact table (the measurable data) in the report.
5. Click **Add Dimension** to select the entities to use as dimensions in the report. This is an optional step. The entities displayed under **Select zero or more dimensions** are the ones that are related to the entity selected as a fact table in the previous screen. You can navigate over the relationships of the entities using the expand buttons.
6. Click **Export selected**.
A file is downloaded.

Alternatively, you can download a Data Object Design file containing all entities of the current and related solutions using the **Export all entities** option.

Notes:

- A To Many relationship is represented as an extra data set with links to the source and target entities.
- If there are multiple To One relationships between two entities, only one is included in the Data Object Design file.
- If there are multiple To Many relationships between two entities, only one is included in the Data Object Design file.
- If there are multiple combinations of To Many and To One relationships between two entities, only one To Many relationship is included in the Data Object Design file.
- If an entity does not have a relationship with any other entity, the corresponding data set is excluded from the linked data model.
- Using renamed or removed entities or properties in a solution will break reports that use them. In this case, use the latest Data Object Design file and modify existing reports accordingly
- EIS entities are not included.
- The contents of the file are within the context of the current organization, regardless of the selected context (Organization or Global).

The package version on which the file was generated is included in the generated Data Object Design file. You can see the version in Analytics Designer in the **Property Editor - Data Object** tab under **General > Title**.

A report designer can use Analytics Designer to build a report on the entities.

To design an entity report in Analytics Designer:

1. In Analytics Designer, import the downloaded file.
2. Complete the report design.

See the *Analytics Designer User Guide* for more information.

Publishing reports to other iHub environments

Reports are not maintained in the Process Platform solution, nor are they packaged with the other artifacts in the Process Platform application. In order to bring reports to another environment, they must be published to the iHub environment manually using Analytics Designer.

The generated Data Object Design file contains a Data Source that holds the database configuration of the environment where the entities have been published or deployed. If you are working with multiple environments, the recommended way to bring reports and dashboards to another environment is to use only the Data Object Design file that is downloaded from the respective environment after deploying the application packages. This file contains the database configuration and SQL queries that will be different for each environment.

Before publishing an Analytics Designer project to another environment, simply replace the existing design file in your project with the file downloaded from the target environment. All

reports and dashboards will automatically point to the new environment, as the name of the design file is the same for all environments. Reports developed on a Process Platform environment will only work on other environments if both use the same database type. The reason for this limitation is that reports refer directly to database column names, which differ based on the database type.

The entity database schema is not identical on every environment the solution is deployed on, as some tables or columns use random strings as part of their name. Therefore, using connection profiles to connect the same Data Object Design file to the different environments is not a good approach. Always use the file that is downloaded from the same environment as the solution is deployed on.

Displaying reports in Process Experience

You can display iHub reports in Process Experience through the iHub panel. The basic steps to enable Process Experience users to view iHub reports:

1. Design a report and publish it to the iHub server.
2. Configure the iHub Connection Manager to inform Process Platform about the location of the iHub server so that it can connect to it.
3. Add a layout that contains an iHub panel to your project.
4. Optionally, configure dynamic filtering for a report added to the layout.

The following sections give details for each task.

To design a report and publish it to the iHub server:

- See the OpenText Analytics documentation for information about designing and publishing reports.

To configure the iHub Connection Manager:

- See [Creating an organization](#).

To configure a layout that contains an iHub panel:

1. Open your project in Collaborative Workspace.
2. Add a layout.
3. Drag an iHub panel onto the layout and configure the general properties such as Name and Chrome.
4. In the Panel Properties, select a report or dashboard.
The reports and dashboards displayed in these lists are retrieved from the iHub volume that has been configured in the iHub Connection Manager.
5. Click  (Save).

Notes:

- Reports with parameters can be displayed only if the parameters have default values or are dynamically filtered using the option described in the following procedure: [To configure dynamic filtering](#). Default parameter values can be configured in the report

design.

- To make the reports display responsively, use the percent unit in Analytics Designer for all report and chart elements. Also set the report layout property to **Auto layout**.
- The following iHub toolbar options are not supported (and are hidden from the menu) by the iHub panel:
 - Parameters
 - Link to this Page
 - Launch Viewer

Configuring dynamic filtering:

The iHub panel enables filtering a report based on the property expressions. For example, you can filter a report on the name of the currently logged in user or the value of an entity property.

Before you begin:

- Be sure that the iHub report you will use has a report parameter.
- Be sure that the iHub folder that contains the selected report does not contain special characters. Otherwise, it cannot read the available report parameters of a report.

To configure dynamic filtering:

1. Add a layout with an iHub panel and select the report.
2. Select **Filter report**.
3. Under **Property**, using an expression, type the property to be linked to the report parameter.
See [Expression language](#).
4. Under **Report parameter**, select the report parameter that will receive the value of the selected property.
5. Optionally, add more filters by clicking (plus sign).
6. Click (Save).

On displaying the layout, the report will be filtered dynamically as the values of the selected properties are passed to the associated report parameters.

Note: In the home page layout, entity properties cannot be used for filtering. Entity properties are available only in the entity layout.

Properties of type Long Text, Date and Time, and Duration cannot be used to filter a report. Cascading or Dynamic filter report parameters cannot be used to filter the report using a property value.

Integrating Capture Center

Document capture and business process management are two technologies that are closely related. Many business processes start with the arrival of a document at an organization. A capture solution such as OpenText Capture Center takes all incoming documents whether arriving as paper mail, fax, email, or any other means, classifies the documents, extracts relevant data from the documents, and feeds them into the appropriate business process. With Capture Center connecting to the Process Suite entity model you have a tightly integrated solution to automate any document driven business processes.

OpenText Capture Center uses the most advanced document and character recognition capabilities available to turn documents into machine-readable information. Capture Center captures the data stored in scanned images and faxes and interprets it using optical character recognition (OCR), intelligent character recognition (ICR), intelligent document recognition (IDR), adaptive reading, and other technologies. As a result, Capture Center reduces manual keying and paper handling, accelerates business processing, improves data quality, and saves you money.

By integrating Capture Center with Process Suite, you can capture and validate data in both electronic and paper documents. If the validation succeeds, an instance of the relevant entity can be automatically created. Once finished, Capture Center creates an instance of an entity that holds the data and has the document in the associated file. Usually the entity will be configured to trigger a process when an item is instantiated.

To integrate Capture Center with Process Suite:

1. In Process Platform Collaborative Workspace, if necessary, create a workspace and project.
2. Create the entity that you want to be integrated with Capture Center.
 - a. Add the relevant properties. These properties will show up as data extraction fields in Capture Center. You specify in Capture Center whether to create a single instance of child properties or whether to create multiple instances (by handling the child properties as a table in Capture Center).
 - b. Add the File building block.
 - c. Add lists, forms, and layouts to customize how items created by Capture Center are displayed in Process Experience.
3. Publish the project to Process Experience.
4. Start the OCC Customizing process to perform the following configuration tasks:
 - Create a new entity.
 - Select Process Suite as the export destination and specify the export parameters.
 - Define the document class to be exported.
 - Test the configuration by importing an image, opening it for validation (either manually or by using rules), and then verifying that an item was created in Process Experience.

See the *OpenText Capture Center Customizing Guide* for complete instructions.

Design considerations

Assume that you have developed a Claim Handling application. Claims come in from various sources, both digitally (such as fax and e-mail) as well as through hard copies.

For the hard copies, you do not want to manually enter the details into the system. Instead, you let Capture Center scan each claim and create a corresponding Claim instance in the system. However, since the details on the hard copy are hand-written, there may be issues with digitizing the details. Therefore, when instantiating a Claim from Capture, a process needs to be started to validate the details of the claim. For example, a check needs to be made to determine whether the claimant is recognized in the system, whether all mandatory fields are filled in, and so forth. You could distinguish claims initiated from Capture Center from claims entered directly in the system by adding a property (such as **Source**) to the Claim entity. The validation process should then be fired only for the claims whose **Source** property contains a value of **Capture Center**.

You also need to validate the digital copies for completeness and accuracy.

If any of the checks in the validation process fail, additional information needs to be requested from the claimant. For these claims, you could introduce a dedicated list such as **Incomplete claims**. Once the incompleteness of a claim has been resolved (either automatically or manually), it can go through the regular process.

After the claim is validated, a new item is automatically created in Process Experience. You can control how the item is displayed using rules, forms, and layouts.

The following example shows an invoice that was created in Process Experience using Capture Center, along with an image of the source paper-based document.

OPENTEXT | Process Experience

Home Page Create new cordys Logout ?

Form

Invoice Date: 15.06.2015 | Invoice Number: 70011 | PO Number: 4500017405

Vendor	Tax
Company Name: C.E.B.	Rate: 3,5
Street: 7890 Broad Street	State: ny
City: New York,	ZIP: 10.001

Create

Line Item

Quantity: 2	Unit: PC	Description: Netbook 10.1 Inch -1 GB RAM -Clear BlackEnterprise edition	Unit Price: 1.340
Quantity: 5	Unit: PC	Description: 23-Inch Widescreen LCD Monitor with Speaker, Shinywhite	Unit Price: 288,4

Web Content

C.E.B. NEW YORK

7890 Broad Street
New York, 10001
Phone : 212-555-1000
Fax : 212-555-1011
Email : info@ceb-newyork.com

Innate Corporation
1280 Lincoln Avenue
New York, NY 10018

Invoice

Date: June 15, 2015
Invoice number: 470011
Salesperson: J. Smith

Purchase order
4500017405

Quantity	Unit	Description	Price	Total
2	PC	Netbook 10.1 Inch -1 GB RAM -Clear BlackEnterprise edition	\$1,340.00	\$2,680.00
5	PC	23-Inch Widescreen LCD Monitor with Speaker, Shinywhite	\$288.40	\$1,442.00
8	PC	External Hard Drive - 2 TB USB 3.0 Ultra	\$89.00	\$712.00

Freight: \$30.00
Net amount: \$4,841.20
Sales tax 2.5%: \$121.04
Total amount: \$5,012.24

Integrating BPM processes and entities

There are two ways to integrate a BPM process and an entity.

- Trigger a BPM process from an entity rule.
 - Use entity web services in a **BPM** process activity.

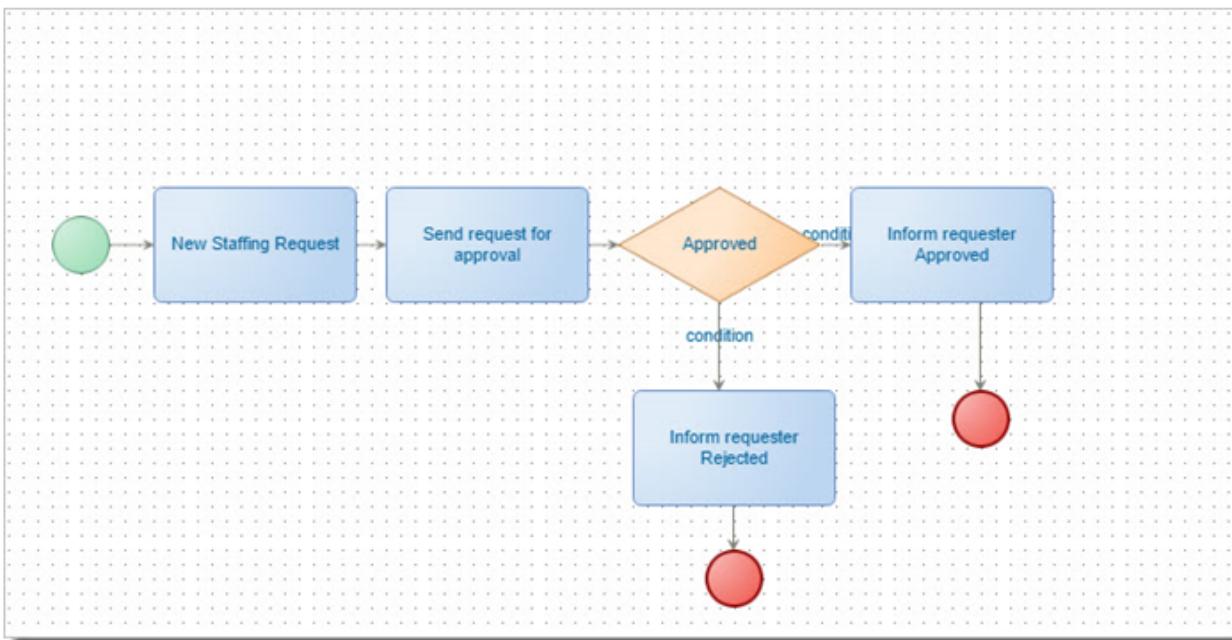
In most cases, the BPMI process that triggers from an entity also needs more entity data. Therefore, it is quite common to have a combined approach.

The basic steps are as follows:

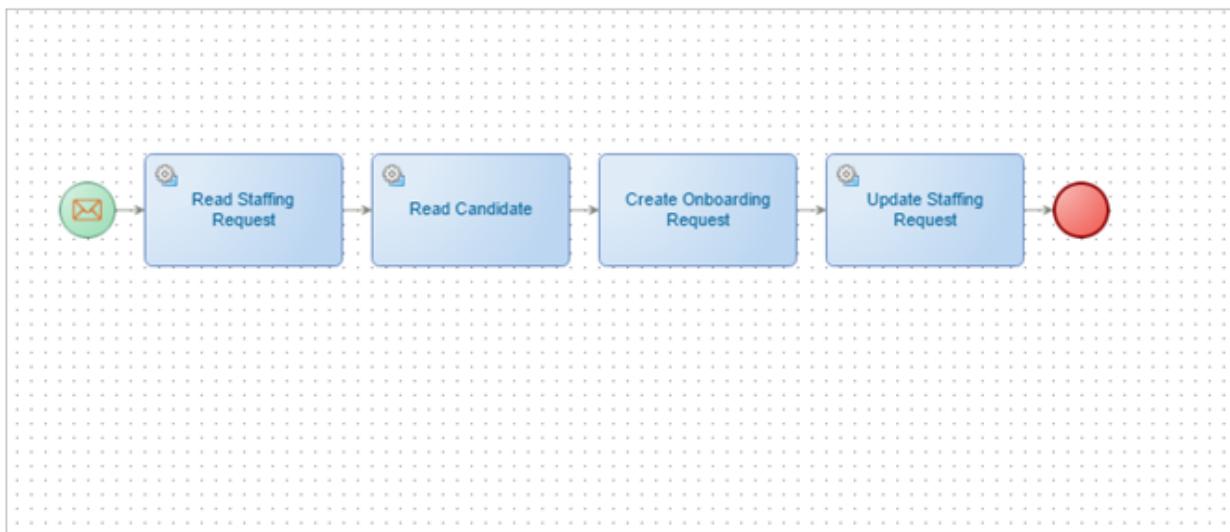
1. Generate web service operations on an entity.
 2. Create the BPM process.
 3. Add the necessary entity web services as activities in the BPM process.
 4. Map the input and output of each activity in the message map.
 5. Create a rule on the entity and select the BPM process that you want to trigger.

In some cases, you will need to take extra steps if you want to connect your BPM processes to entities. For example, if you want to create a multi-level approval process that is triggered by your BPM process. The following examples show several BPM processes for a staffing application.

The following diagram shows the Staffing Request process.



The following diagram shows the Employee Onboarding process that has web services attached to activities.



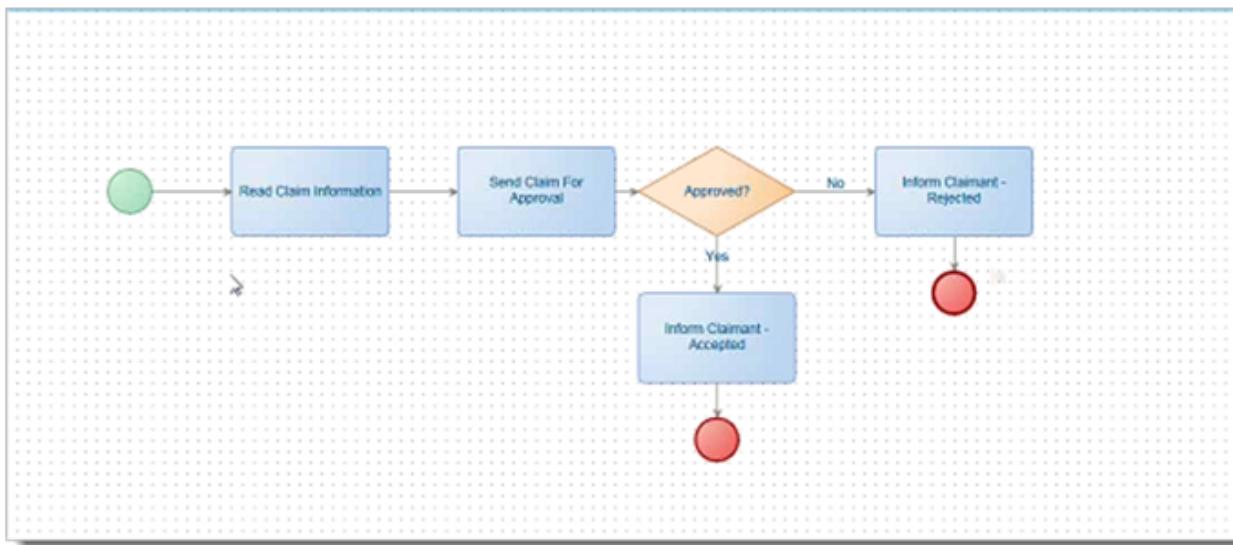
Process overview

1. Create a new workspace and project where you plan to build an application using Entity Modeling.
2. Create the BPM process model.
3. Create a rule on the entity.
4. Import web services.

5. Modify your BPM process model to connect the web services to the activities.
6. Publish and review changes in Process Experience.

Creating the BPM process model

When you build an application using Entity Modeling, you might also want to include various process models. For example, you could have one larger model to show the entire functional and technical requirements or several small models, such as this one, for an approval process.

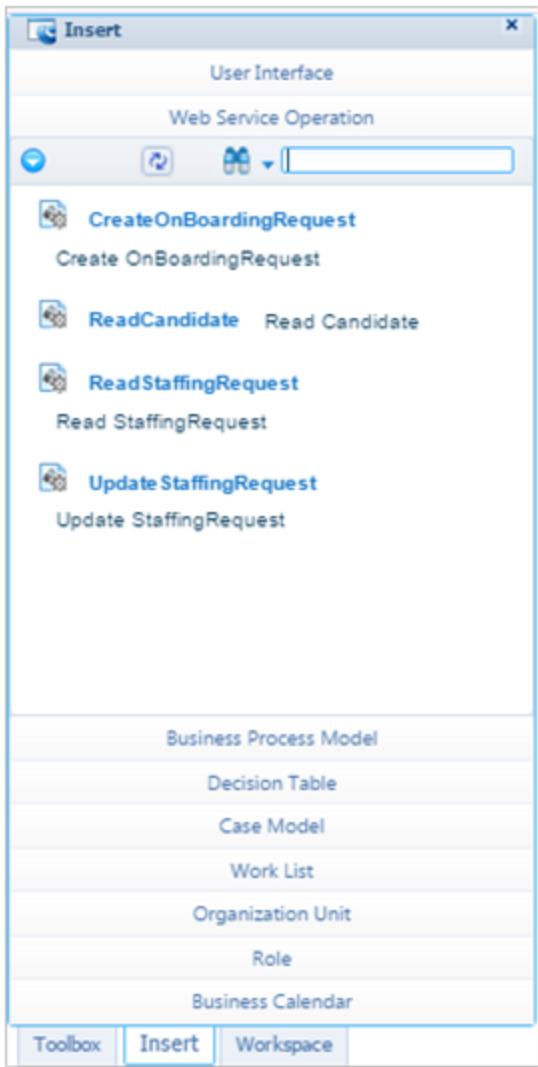


You can integrate your BPM process with your entity application in a variety of ways. For example, you can create a multi-level approval process. Creating a business process is easy but, if you need help, see the Process Platform online help.

Follow this basic process:

1. Open the project that includes your entities.
2. Add a new business process. (Click **New > Other > Business Process Model**.)
3. Bind the entity web service operation to an activity in the business process model.

There are several ways to bind an entity web service operation to an activity. You can click the Insert tab on the left to display the Insert pane or right-click and choose **Insert > Web Service Operation**. The following figure shows the Insert pane and entity web services that are available (in this scenario) to bind to an activity.



4. When finished, save the process.
5. Keep the business process open and return to the workspace.

Next, you need to create a rule on the entity that initiates the process.

Note: When you create the BPM process, you can review the namespace, but you should not change it.

Creating the rule

If you have processes in your project, you connect your entity to a process by creating a rule. The rule will take action, based on the conditions, and then start the process.

See [Adding rules to an entity](#) for complete details.

The following example shows a rule that triggers a process when an action is initiated. In this case, when a candidate accepts a job offer, the employee onboarding process is triggered.

Rule Properties

When:

Action ▾

If:

Basic Advanced

all ▾ of the following are true:

Status equal to Offer Accepted ▾

Then:

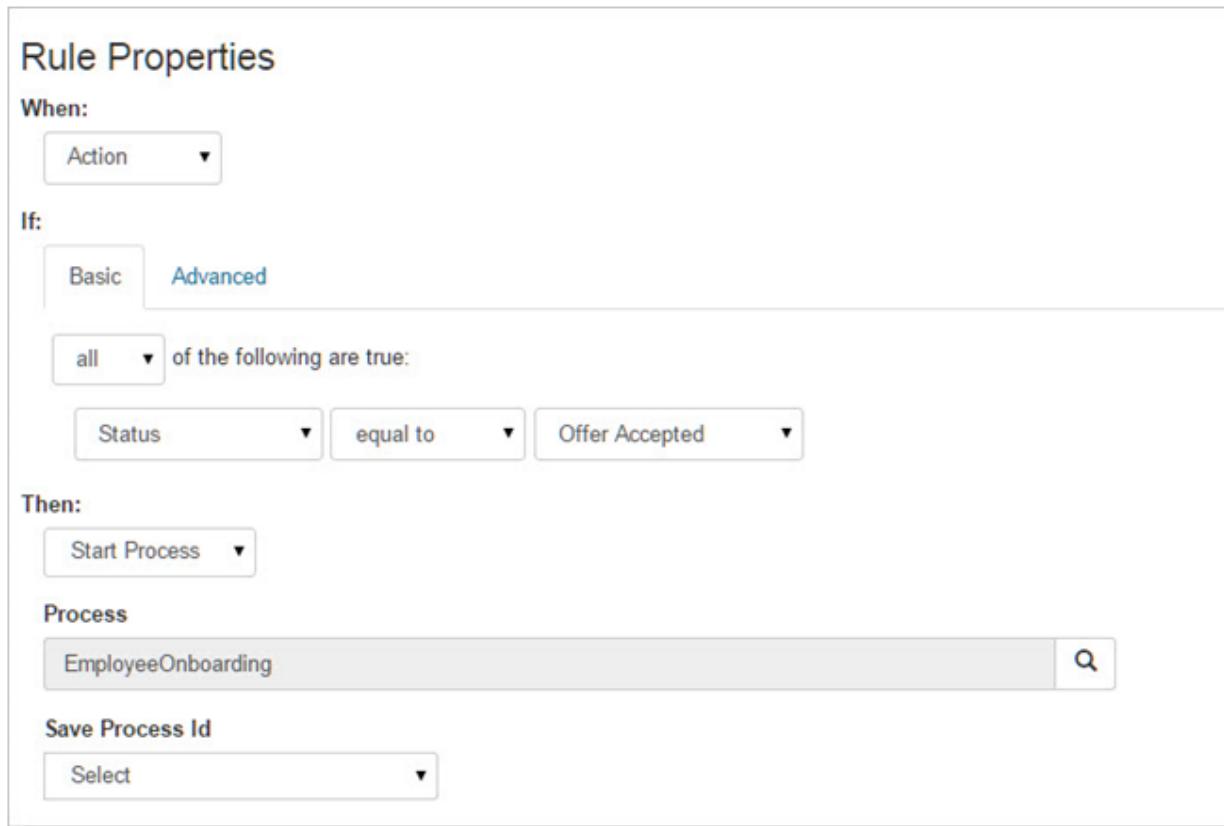
Start Process ▾

Process

EmployeeOnboarding

Save Process Id

Select ▾



Follow this basic process:

1. Open the entity for which you want to add the rule to start the process.
2. Add a rule.
3. Define the When and If rule properties.
4. Select **Start Process**.
5. Click Browse to select the process.
6. Click  (Save).
7. Close the Rule Properties and return to your entity.
8. Add web services to the entity.

Adding web services to the entity

After you create the process and a rule that calls the process, you need to enable web services on the entity.

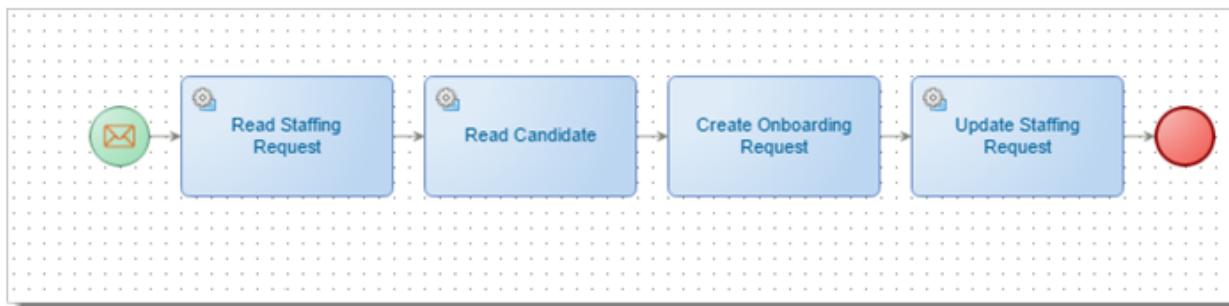
1. If necessary, open the same entity that you added the rule to call the process.
2. Click **Add > Web Service**.
3. Select the basic operations. For example, in a staffing application, after a new candidate has accepted the job, you could initiate an employee onboarding process. HR staff members need to be able to view information to determine where things are in the process. In this case, the Read basic operation on the web service is required.
4. Click **Add**.

Adding web services to the process

Next, you can add the web services to the appropriate activities in your business process.

Note: If you have rich text content in your long text property, and you want to copy (map) it to another long text property, make sure that the content is handled as text. You do this by clicking the property name and specifying 'Use:' = 'Replace XML as String' 'Select with Target NS'"

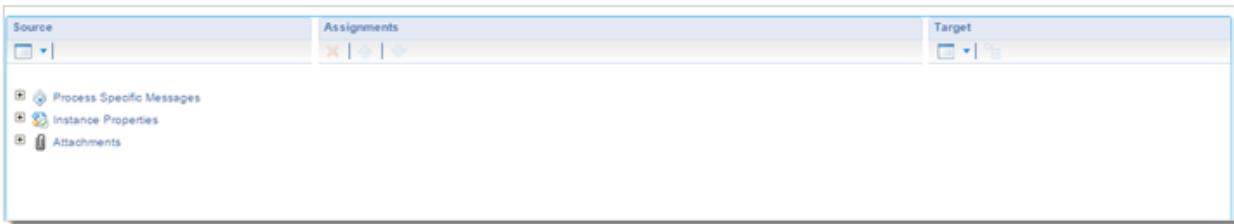
1. Open the process. The Toolbox automatically opens on the left.
2. Click the Insert tab.
3. Open the Web Service Operation section.
4. Drag the web service from the workspace area to the activity in the process.
5. Edit the description.
6. Repeat this process as necessary. The following example shows the employee onboarding process with web services added to three of the activities.



7. Save the process.
8. To modify the message maps, click the Message Map tab. This step is required to complete the entire process to connect your process to the entities and web services.

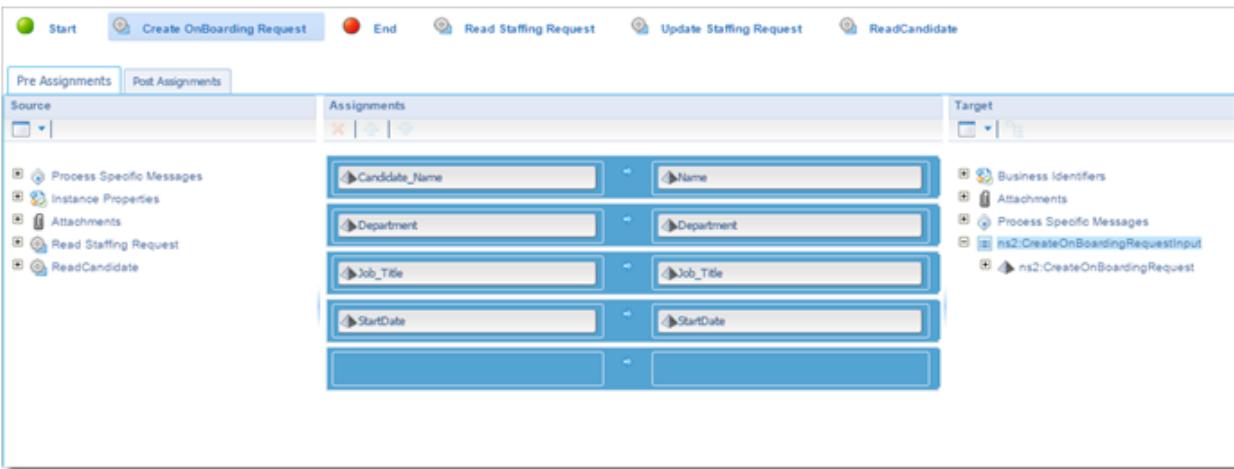


The message map shows three panes: Source, Assignments, and Targets.



9. Select the activity that you added the web service. For example, Create OnBoarding Request.
10. Drag the source and target that you want to the Assignments pane in the middle.

The following example shows the assignments for the Create OnBoarding Request.

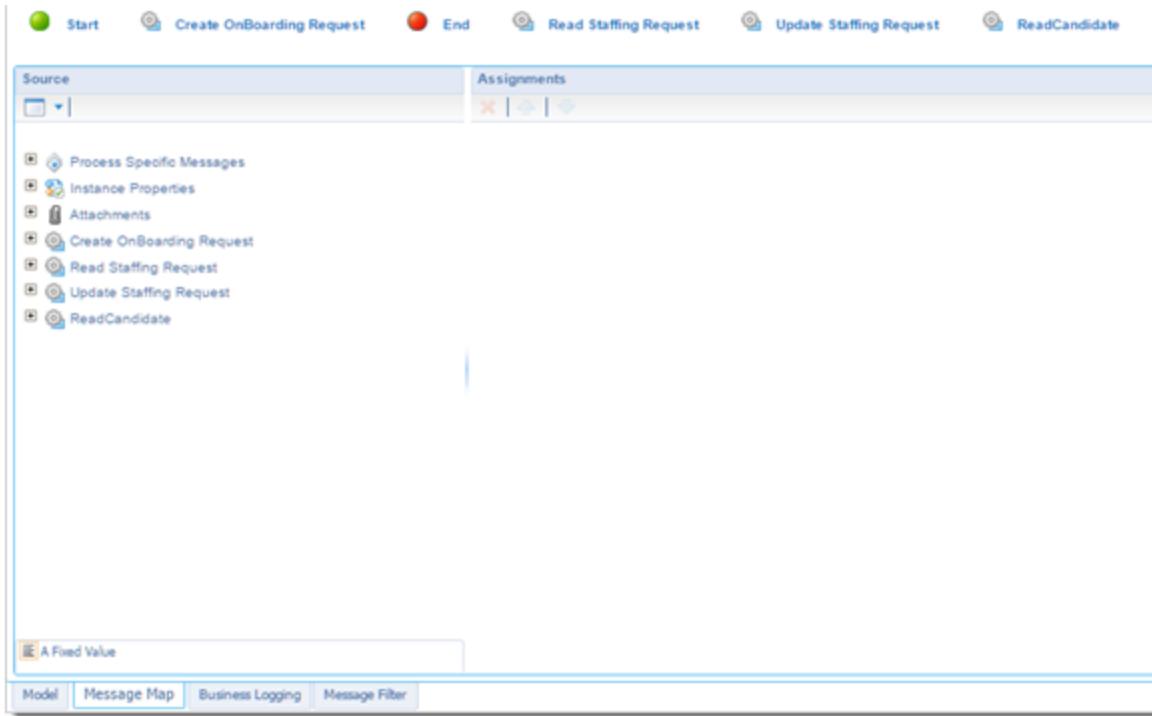


For specific information on creating assignments, see the Process Platform online help.

11. Repeat the process for the message maps for each activity. Then, save and close your process.
12. Return to your workspace and publish.
13. If necessary, resolve any errors.

Creating an input message

1. In your BPM process, select the Message Map tab.



2. In the Source section, right-click **Process Specific Messages** and select **Create Message**.

The name of the message syntax must be <entity-name>-id. For example, Candidate-id.

3. Right-click this message and select **Create Element**.

4. Give this element the name **Id**.

This message and element define the XML that will be sent from the entity runtime to the BPM process when it is triggered.

5. Follow these steps to complete the message map for each activity.

For additional information on message maps see the Process Platform online help. Also see [Dynamic enumerations](#).

Configuring multi-tenancy support

Entity-based applications can be deployed to multiple tenants (organizations). For example, you can create a separate tenant for departments or customers. All tenants can use the same database or they can use different databases.

- An application that is deployed specifically to a particular tenant uses a tenant-specific database to store items. That tenant's data is kept separate from that of other tenants.
- An application that deployed to a shared space does not use a tenant-specific database to store items. Each tenant's data is not kept separate from that of other tenants.

The instructions for setting up multi-tenancy refer to the following organizations:

- The **system organization** is where the application resides.
- The **target organization** is the tenant to which the application will be deployed.

To set up multi-tenancy:

1. Set up the target organization. See [Creating an organization](#).
 - a. Assign the sysAdmin role so that you can create a new target organization.
 - b. Create the target organization.
 - c. Select the user who will administer the target organization.
 - d. Assign security to the administrator of the target organization.
2. Optionally configure a separate database for the target organization. You do not need to perform this step if multiple target organizations will use the same database and you know its name. See [Configuring an entity runtime database per organization](#).
3. Deploy the application to the target organization. See [Deploying the application](#).
 - a. Verify that the settings for the security certificate are set appropriately, and set them if necessary.
 - b. Authorize the target organization administrator to manage packages.
 - c. Set package properties and create the application package.
 - d. Deploy the application.
 - e. Assign security for the application.
 - f. Test the application in Process Experience.

For more information about each step, see the Process Platform Product documentation, particularly the following topics:

- Multi-tenancy
- Authorize organization administrator to manage packages
- Setting properties of an application package
- Creating and downloading application packages
- Deploying applications using Application Deployer

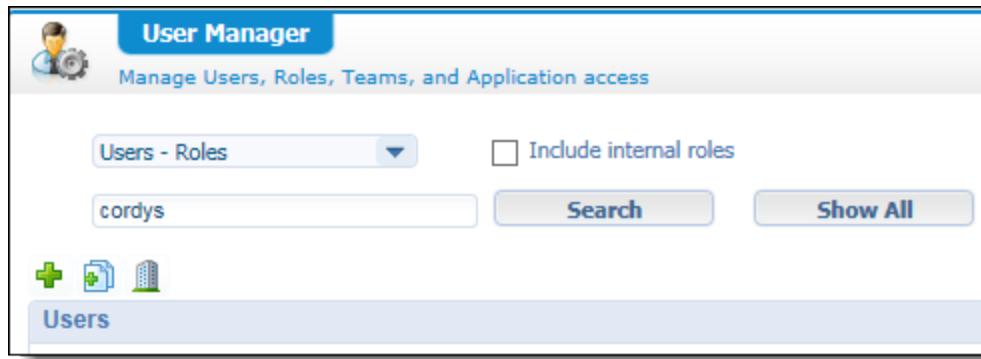
Creating an organization

An organization serves as a container where you can create and manage entities required to work with Process Platform. These entities can be users, functional components, and user interfaces. An organization provides a unique space to develop your custom application. Organizations can be treated as virtual separators of various business divisions within an enterprise.

To create an organization, you must have the role of systemAdmin.

To assign the systemAdmin role:

1. In the system organization, start the Process Platform User Manager.



2. If an existing user will administer the target organization, assign systemAdmin to that user.
3. If you want to create a new user to manage the target organization, click **+** (Add a User) and configure the user as follows.

In this example, the User Name is Acme Administrator but you can use a different name.

The 'Create User*' dialog box contains the following fields:

- Authentication Type:** Options include External (radio button), Cordys (radio button, selected), and Certificate (radio button).
- User Name:** Acme Administrator
- User Full Name:** Acme Administrator
- User ID:** acmeAdmin
- Default Organization:** System

A section titled 'Assign Password' contains:

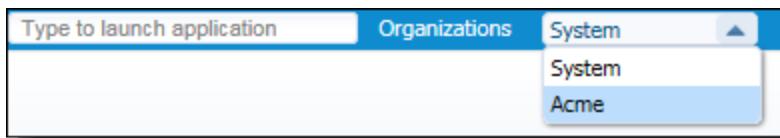
- Password:** [REDACTED]
- Confirm Password:** [REDACTED]

4. In Authentication Type, select **Cordys**.
5. Click **+** (Save).

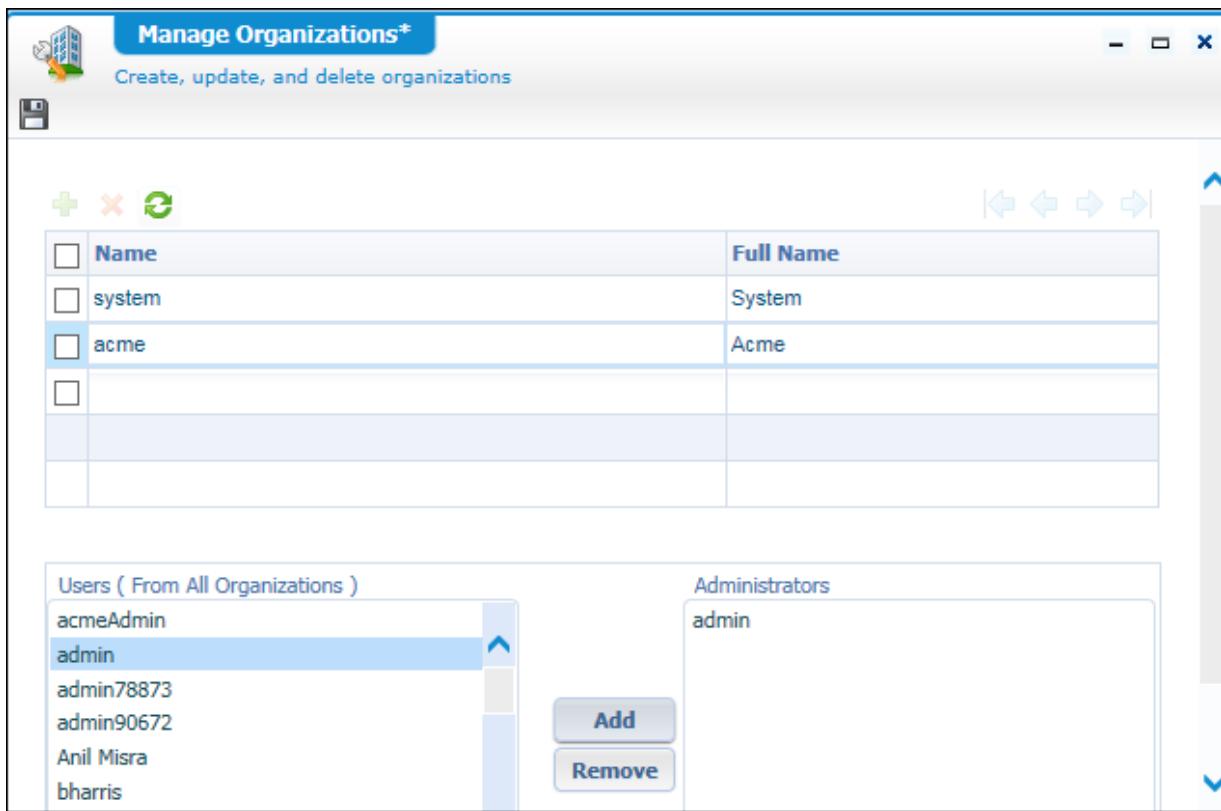
To create a target organization:

1. In the system organization, start the Process Platform Organization Manager.
2. Click  (Insert).
A row is added to the table.
3. Type a Name and Full name for the target organization.
Organization names should not include spaces.
The full name of an organization is editable only after the organization is created.
4. In the Users list, select one or more users who will be administrators of the target organization.
The Users list displays the names of all users from all available organizations.

Tip: A user who is added to the list can switch to the new organization from the list at the top of the Process Platform desktop instead of entering a different URL.



5. Click **Add**.



Name	Full Name
system	System
acme	Acme

Users (From All Organizations)

- acmeAdmin
- admin**
- admin78873
- admin90672
- Anil Misra
- barris

Administrators

- admin

Add **Remove**

6. Click  (Save) and close the window.
The target organization is created.
7. Verify that you can sign in to the target organization.

To select the user who will administer the new organization:

1. In the target organization, open Process Platform User Manager.
2. Select the user who will administer the target organization.
3. Select **Include internal roles**, since you also need to assign the entity runtime roles.
4. In the Roles list, search the following roles and drag them to the user:
 - Administrator
 - Developer
 - Entity Runtime Administrator
 - Entity Runtime Developer
 - Entity Runtime User

Enabling Process Experience viewers to access reports

If you plan to have your users access iHub reports and dashboards, you must configure the iHub Connection Manager, which enables connecting to a separate iHub volume for each organization.

Tip: To prevent users in one organization from accessing data that belongs to another organization, the recommended configuration is to use a different iHub volume for each Process Platform organization.

To perform this procedure, you must have the Administrator role.

To configure the iHub Connection Manager:

1. Verify that the volume was already created on iHub and, if not, create it.
2. Sign in to Process Platform in the new organization.
3. Open the iHub Connection Manager.

Note: If the system administrator has configured a shared iHub volume for all organizations, the configuration is displayed and you need to overwrite it with one specific for the new organization.

If the target organization should not be connected to iHub, clear (uncheck) **Connect to iHub**. With this setting, no connection will be made from the organization to iHub. As a consequence, related features such as the iHub panel cannot be used in the organization.

4. Configure the URL and volume of the iHub server you are connecting to.
5. Click **Save**.

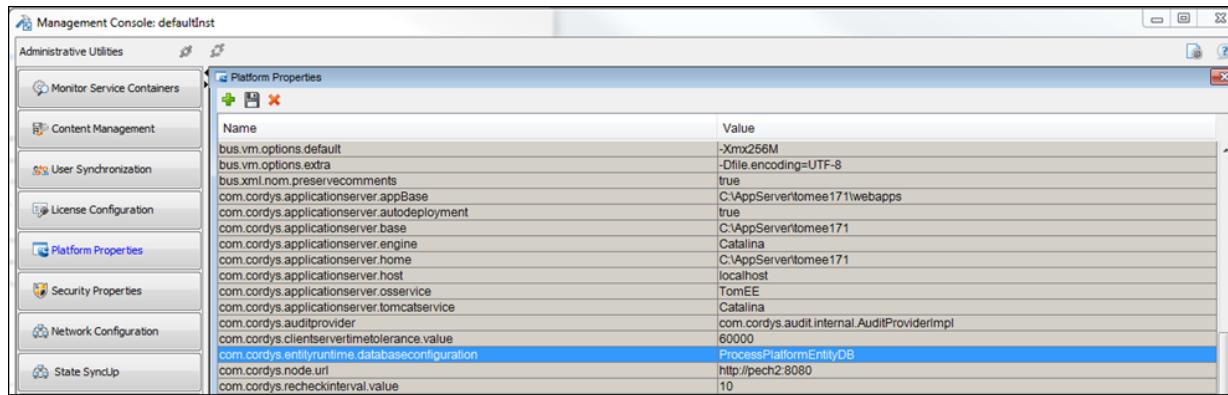
Configuring an entity runtime database per organization

An application that is deployed to a target organization can use an organization-specific database to store entity instances.

To find the name of the database configuration used by the entity runtime:

Note: If you already know the name of the database you can skip this procedure and go directly to configure the target organization to use a separate database.

1. From the Windows start menu, click **Start > OpenText Process Suite Platform 16 > <your instance name> > Tools > Management Console**.
2. In the left menu, click **Platform Properties**.
3. In Name, find `com.cordys.entityruntime.databaseconfiguration` and note the value for the property (for example, `ProcessPlatformEntityDB`).



To configure the target organization to use a separate database:

1. Sign in to the target organization as an administrator.
2. Start System Resource Manager.
3. Click (Manage Database Configurations) on the toolbar of the System Resource Manager window.
4. Click (Insert).
A new row is appended to the table and a section for database configuration details is displayed in the bottom pane.
5. In Name, type the name of the database configuration.
This is the name you noted earlier (for example, `ProcessPlatformEntityDB`).
6. In Description, type a description of the database configuration.
This is optional.
7. Select **JDBC** to enable the entity runtime to connect to the database and enter the connection details.

The screenshot shows the 'Database Configuration' dialog. It has two tabs: 'JDBC' (selected) and 'OLEDB'. Under 'JDBC', the 'Driver' is set to 'Microsoft SQL Server', 'Driver Class' is 'com.microsoft.sqlserver.jdbc.SQLServerDriver', and the 'Connection String' is 'jdbc:sqlserver://pech2:1433'. Under 'Description', the 'Default Database' is 'acmedb', 'DB User' is 'sa', and 'Password' is masked. A 'Save' button is located at the bottom right.

8. Click (Test Connectivity) to verify that Process Platform can connect to the database with the specified parameters.
9. Click (Save).

A new database configuration is created and the name, description, and organization under which the database configuration is being created are displayed in the new row.

Note: For a new user in Oracle, only Create permissions are granted to the user. For other permissions, you must connect to the Oracle server using the client and select the permissions explicitly.

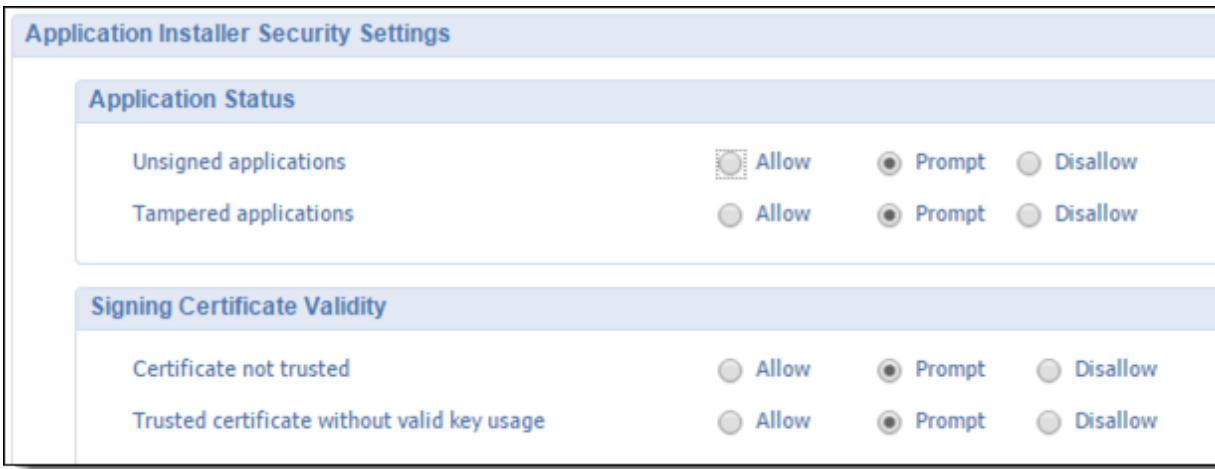
Deploying the application

Application deployment consists of multiple steps, as described in the following sections.

To verify the security certificate settings:

Note: If the settings for the security certificate were already set, you can skip this procedure and go directly to the procedure to authorize the target organization administrator to manage packages.

1. In the system organization, open Security Administration.
2. Select the **Code Signing** tab.
3. In the **Application Installer Security Settings** section, select **Prompt** or **Allow** for the following settings:
 - Application Status > Unsigned applications
 - Application Status > Tampered applications
 - Signing Certificate Validity > Certificate not trusted
 - Signing Certificate Validity > Trusted certificate without valid key usage



To authorize the target organization administrator to manage packages:

1. In the system organization, open Application Deployer.
2. Click  (Configure Authentication) at the top of the Deployment Overview panel.
3. In the Set Authorization panel, click Browse, select the target organization, and then click **OK**.
4. In Authorization Type, select **Full Access** and then click **Authorize**.
5. Minimize Application Deployer.

To set package properties and create the application package:

1. In the system organization, open Workspace Documents
2. Right-click your application and select **Packaging > Package Properties**.
3. In Supported Deployment Spaces, select **Shared** and **Organization** and then click **OK**.
4. Right-click your application and select **Packaging > Create Package**.
5. When the operation completes successfully, click **Download Package**.
6. When prompted, save the package (note where it was saved as you will need this information when you deploy the application).
7. Click **Close** when the download is complete.

To deploy the application:

1. In the system organization, open Application Deployer.
2. In Space, click  (Browse).
3. In the Organizations List, select the target organization click **OK**.
4. In Upload Packages, click Browse, select the application, and click **Open**.
5. Click **Upload and Deploy**.
6. Click **Next** at each Application Deployer page.
7. Click **Finish** when the deployment is complete.

To assign security for the application:

1. In the target organization, run the Administrator tool.
2. Click CTRL+F5 to clear the cache and refresh the display.
3. In the **Workspaces** list in the left pane, select your solution.
4. In the **Configurable Elements** list in the middle pane, expand **Solution Security** and select **Use Solution**.
5. In the **Roles** list in the right pane, select the following roles:
 - Entity Runtime Administrator of OpenText Entity Runtime
 - Entity Runtime User of OpenText Entity Runtime
 - Entity Runtime Developer of OpenText Entity Runtime

The solution is now available for every Process Experience user.

To test the application:

- In the target organization, open Process Experience.
- Create a few items.
- Open Process Experience on the system organization and verify that the items you created are not shown.

Advanced domain modeling using subtyping

When modeling business domains, you are likely to run into situations where distinct entities in the business domain have certain attributes or behaviors in common. Instead of repeatedly modeling these things for each entity separately, a common domain modeling practice is to gather the commonalities in a single "general" entity and model only the things that are specific to each individual entity as part of another "specific" entity. The specific entities can then be specified to inherit the common aspects from the general entity.

The general entity containing the common aspects is often called the supertype, whereas the different entities capturing the specific aspects are called the subtypes. The act of pushing the common aspects to a general entity is called generalization, whereas the act of modeling the specific aspects in separate entities is called specialization. By having the common aspects only at one place, the effort to implement those parts of the application, maintain it, and possibly even extend it in future versions of the application is much less compared to performing these tasks when the common aspects have been duplicated in many places.

For example, consider various types of insurance, such as life and automobile. Typically, for each type of insurance, different pieces of information need to be stored. For both car and life insurance, you would keep track of when the insurance begins and ends. You may also want to send a renewal notice to the insurance owner when the insurance is about to expire. However, calculating the premium for auto insurance is much different than it is for life insurance. For auto insurance, the premium depends on the type of the car (make, model, and year), the accessories added to the car, the mileage per year, and so on. For life

insurance, the premium depends on the age of the insured, lifestyle choices (such as a smokes, drinks alcohol, participates in dangerous sports), and other factors.

If an entity inherits aspects from another entity, the behavior of that entity can then be said to be the aggregate of the behavior specified on the supertype and the behavior specified on the entity itself. In many cases, specializations add things on top of what they inherit from the supertype entity. However, there can also be situations where a particular piece of behavior specified on the supertype needs to be adapted to match the context of the subtype. In these cases, the subtype can override that part of the supertype.

When designing your domain model, it is important to consider how you can use subtyping for maximum efficiency. From a data perspective, subtyping imposes a generalization/specialization hierarchy in the domain model. Therefore, think about the level where you should create properties and relationships and the behavior (or presentation) for each level. Behaviors that are common for all subtypes should be specified at the supertype level.

All building blocks that are added to an entity are inherited by all subtypes of that entity. For each subtype, you can add more building blocks as appropriate. Some building blocks can be added only once to an entity. If a building block has already been added to the supertype, you cannot add the same building block to any of its subtypes.

During the design process, a building block (with the same name) can be added to a subtype entity and later to its supertype. Because building blocks are unique, based on its name, a conflict could occur. Typically, this type of conflict could occur when upgrading a base application. See “Upgrading the base application” for more information.

You can prevent users from creating instances of an entity by selecting the **Abstract** option in the properties for the entity. If this option is selected, the **Create new** menu does not provide the entity as an option and the Create web service operation is not available and cannot be enabled. If the entity is designated as **Abstract** after the web service Create option was enabled, you get a validation error when you publish or package the project. Other web service operations remain available.

Typically, the **Abstract** option is selected for a supertype entity (such as Contract) that contains general properties that are applicable to all contracts. The supertype entity also contains subtype entities (such as Employment Contract or Sales Contract) that contain specific properties that are relevant for a particular type of contract. In this scenario, Process Experience users create instances of the subtypes rather than the supertype.

See [Web services on entities](#) and [Adding forms](#).

To designate an entity as a supertype:

- Select the Abstract option in the entity's properties. See [Adding properties](#).

To create a subtype of an entity:

1. In **Workspace Documents**, right-click the entity and select **Create subtype**.
2. Click  (Show/Hide Entity Properties) and define the properties of the subtype.

Note: The name of the supertype is displayed automatically and cannot be changed.

3. Configure the building blocks that are appropriate for your application.

It is helpful to create a folder structure to help you quickly identify subtypes of a particular entity.

You can also subtype entities that are part of other applications. To do this, you need to import the model package of the application that contains the entity into your development workspace and perform these steps on the entity to be subtyped. The steps to create the model package of an application are described in [Setting Properties of an Application Package](#) in the Process Platform product documentation.

Note: If the supertype of a subtyped entity is removed, an error occurs when you try to validate or publish the entity. To resolve the error, you must delete the subtyped entity.

Replacing building blocks

If the configuration of the building block on the supertype does not match the desired behavior on the subtype, you can override the building block by replacing it and then configuring it to make it meet the requirements of the subtype. If there is no longer a need to replace a building block in a subtype, you are able to undo the replacement and revert back to the behavior as it was specified on the supertype.

To replace a building block:

1. Open the subtype entity.
2. Point to the building block to be replaced.
If the building block is replaceable, a **Replace** button is shown in the highlighted bar around the building block.
3. Click **Replace**.
4. Configure the building block based on your requirements.

Important: The following building blocks cannot be replaced:

- Assignee
- Deadline
- Discussion
- Identity
- Property
- Relationship
- Tracking
- Mobile app

There are a number of building blocks that are not inherited from an entity's supertype and that are not shown on the entity subtypes. If you want to use these building blocks on any of the subtypes, you can add them. This applies to the following building blocks:

- Business Workspace
- Lists

- Mobile app

Replacing forms

A form can include one or more other forms, defined either in the same or in related entities (if the entity has relationships). This also applies for subtyping. Forms of the supertype are available on the subtype for reuse. See [Nesting forms](#) for details.

Replacing email

To replace the email configuration of a subtyped entity, replace the email building block and select a new email configuration.

Conflict detection

The system handles conflict detection and displays an error message when a conflict occurs. Generally, a conflict occurs when information that was previously changed in the subtype is updated in the supertype. A few examples of conflicts follow:

- A property is added to the supertype but a property with the same name was already added to the subtype.
- A property that is used in a subtype is deleted, renamed, or replaced in the supertype.
- A form that is used in a subtype is modified or renamed in the supertype.

The changes that you need to make to resolve the error will depend on the type of conflict. You can either retain the changes in your subtype or you can accept the changes in the supertype (which may require modifications to the subtype).

Considerations for using subtyping

This section provides some information you should consider if you plan to apply subtyping within your business domain. The insurance example described previously in this topic will be used here as well.

Identify generic elements.

The generic characteristics common to both insurance types (car insurance and life insurance) should be included in the Insurance entity. For example:

- Properties: summary, premium, start and end dates
- Relationship: owner entity
- Forms: create, default
- Lists: all contracts, expired contracts
- Rule: send expire notification 1 month before expiration (based on end date)

Identify specific elements

For car insurance, you might consider the following specific characteristics:

- Properties: yearly mileages
- Relationship: car
- Rule: premium calculation

For life insurance, you might consider the following specific characteristics:

- Properties: smoking cigarettes, drinking alcohol, doing dangerous sports
- Rule: premium calculation

Integrate specific elements into existing building block configurations

Because the specific elements have additional properties, include those in the forms and lists of the supertype by replacing them within the subtype and add them to the forms and lists.

Will the supertype be used on its own?

If not, it should be marked as Abstract to prevent users from creating instances of the supertype.

Defining security for a customization or subtype

After customizing the functionality of an application by adding properties, relationships, or other building blocks, you need to define security for this functionality by customizing the Security building block of the entity.

If a base entity is subtyped, the permission on this subtype can be set. The security as defined on the base entity is available and can be overwritten.

There are basically two rules that can help you redefine permissions in the Security editor:

- Building blocks defined on the supertype or original entity are displayed in gray. The newly added building blocks are shown in black. This enables you to quickly see whether a change in permission was made on the customization or whether the permission as defined on the base was overwritten.

For example, assume that an entity called User has properties called Name and Address and then it is customized by adding a new property called Phone Number. In the Security Editor, the Name and Address properties are shown in gray but the Phone Number property is shown in black.

- When customizing a permission that is defined on the base entity, a Revert icon is shown next to that permission in the Security Editor. You can use this icon to revert this specific security customization.

For example, building on the previous example of the User entity, assume that for a role on the base entity Read permission is defined but no Update permission is defined. After

customizing the Security building block by adding the Update permission, a revert icon will be shown next to the Update permission. This indicates that this specific permission was customized. Clicking the revert icon remove the customizations and revert the state of this specific permission to the value as defined on the base entity.

This rule also applies for configuring permissions of a subtyped entity.

Creating a theme

Using the Theme Designer, you can create a custom theme for a shared space or organization in Process Experience.

You can customize the following components:

- Header bar
 - Logo image

Tip: In addition to creating a theme, the logo image can be customized in other ways.

See [Customizing the logo](#) for a description of each method.

- Panel Canvas (background color)
 - Background gradient top
 - Background gradient bottom
- Panels
 - Panel header background color
 - Panel border color
- Actions
 - Action bar button background
 - Action bar button color
 - Button Radius

A Preview option is available so that you can view the theme before publishing the project. You specify the URL to the layout where the preview is shown. By default, the URL is the Process Experience Home Page layout.

The designed theme is published to the organization/shared folder when the project is published. This overwrites any previously-defined theme. When Process Experience runs, the current organization theme is applied. Each organization can have only one active theme. If the organization has no theme defined, the shared organization theme is used. If the shared space has no theme defined, the default Process Experience styles are retained.

Note: Using most browsers, a color picker is available from which you can select colors. If you use Internet Explorer, colors must be specified as hexadecimal values.

To create a theme:

1. In **Workspace Documents**, select a project and click  (New) > **Other**.
2. Search for theme and select it from the results.
When the theme editor is first loaded, the Process Experience default configuration values are shown.
3. On the Theme Editor tab, define the following options for the theme:
 - Background gradient top/Background gradient bottom - The top and bottom colors for the gradient of the layout canvas.
 - Font - The font used within Process Experience.
 - Panel border color - The color of the border for panels.
 - Font color - The color of the font used within Process Experience.
 - Button radius - The "roundness" of the action buttons: 0px = square, 16px = rounded
 - Panel header background - The color of the panel headers when using full chrome layout options.
 - Action bar button background - The background color of the action buttons.
 - Action bar button color - The color of the action button text.
 - Preview URL - The URL where the theme will be previewed.
4. If you want to replace the existing header logo, on the Edit Banner tab, click **Select Files** and select an image.
The recommended dimensions for the image are 55px in height and 385px in width.
5. If you want to see the theme before publishing, click **Preview**.
6. Click  (Save).
 - a. Specify the Name, Description, and Location for the theme.
 - b. Click **OK**.
7. Publish the project.

To restore the default image for the banner:

1. Navigate to the .empower-logo class from the custom.css file.
 - For Windows, the custom.css file is in <drive>\Program files\Opentext\Process Platform\defaultInst\organization\<organizationname>\themes\custom.css
 - For Linux, the custom.css file is in opt\opentext\process_platform\defaultInst\webroot\organization\<organizationname>\themes\custom.css
2. Delete everything between the following curly braces "{}" and "{}"
By default this is line 21 to 23 of the custom.css file

Removing a custom theme

When a custom theme is created, a `custom.css` file is created within the Process Platform web content directory for the organization.

To remove the custom theme for the organization you can use either create and publish a new custom theme or you can remove the `custom.css` file from the file system.

To remove the `custom.css` file from the file system:

1. Navigate to the `custom.css` file.
 - For Windows, the file is in `<drive>\Program files\Opentext\ProcessPlatform\defaultInst\organization\<organizationname>\themes\custom.css`
 - For Linux, the file is in `opt\opentext\process_platform\defaultInst\webroot\organization\<organizationname>\themes\custom.css`
2. Create back up of the `custom.css` file and then delete it from the directory.

Chapter 15

Tips and tricks

This section provides guidelines and instructions for performing specific functions in Entity Modeling.

Working as a team

If you plan to work as part of a team, you must add the project to source control. There is no easy way to do this after the project is created and development is in progress. You must do it from the beginning when you create a workspace. Multiple team members cannot work on the same entities simultaneously so it is important to collaborate with your team members to plan which parts of the application each team member will work on.

For example, before you begin developing a claims application, you might define the direction for team members similar to the example displayed in the following table.

Team Member	Area
Jim	Claims
Jennifer	Members
Liam	Organization
Karen	Person
Ben	

Before you begin, you need your SVN UCommonRL, user name, and password to access the SVN server.

To connect to SVN:

1. Create a new workspace.
2. Enter the name and description.
3. Under Source Control Management type, select **SVN for team development** and then click **Next**.
4. Type the URL to the SVN server. (Alternatively, copy and paste the URL from SVN Repo-browser.)
5. Type your user name and password credentials to access the SVN server.
6. Click **Test Connection**.
If the connection is successful, an information message is displayed.
7. Click **Close** and then click **Next**.
8. Type the project name, package owner, and leave the product name that is automatically populated. Then, click **Next**.
9. Click **Create**.
10. After the workspace is created, click **Close**.

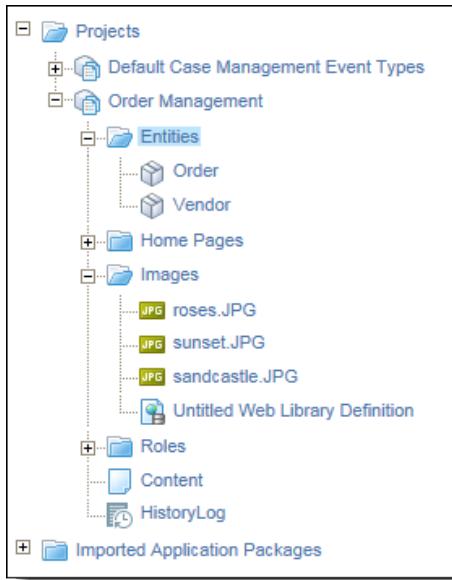
Organizing your project

It is good practice to create folders in which you can organize the various components of your project. As your project grows, you may have many entities, roles, home pages, images, and so forth and organizing them in folders will make this information easy to find.

To create a folder:

1. In **Workspace Documents**, right-click your project and select **New > Folder**.
2. In the text box that appears, type the name of the folder.
3. Drag the relevant content to the folder. You can also drag the folder to another location.

In the following example, separate folders were created for Entities, Home Pages, Images, and Roles. You can collapse and expand folders by clicking the plus or minus sign.



Developing lists of master data

You can create entities that enable an authorized user to maintain lists of various types of master data directly from Process Experience. A single master data list can be shared by multiple entities (such as Customers, Employees, Vendors, or Products).

For example, an entity called Country can contain a list of countries that Process Experience users browse to select a country when creating a Vendor. This frees users from having to enter the country directly each time they create an item and also ensures that the country information is entered consistently, regardless of the type of item.

Before you begin:

If you do not already have an entity that you can use to create an item in Process Experience, create one and populate it with the required building blocks. This is the entity that you will use to test the shared master data. The following procedure uses Country as an example of master data and Employee as an example of an application that access the master data.

To create a Country master data entity:

1. Create an entity called Country. See [Creating entities](#).
 - Add text properties called Country (do not change the default length) and Country Code (set length to 2). See [Adding properties](#).
 - Add a list called All Countries and include the Country and Country Code properties. See [List](#).
 - Add a Create form that includes the Country and Country Code properties. See [Adding forms](#).

2. Open the Employee entity.
 - Add a To One relationship called Country and select Country as the target entity. See [Adding relationships](#).
 - Edit the Employee entity's Create form to include the Country property and folder. Select **Show Browse** for the icon and **All Countries** for the list.

Tip: The drop list presentation on a to-one relation works very well with master data entities that don't have a large number of choices.

3. Publish your workspace.

To test the master data in Process Experience:

1. Open Process Experience and refresh the display.
2. Select **Create new > Country** and create a few country items.
3. Create a new employee and use the Browse button to select a country.
4. Open the employee item that you created and notice that the country is displayed. You can change the country by clicking **Browse**.
5. Open the All Countries list. You see a list of all the countries and country codes that you created.

Creating lists to use with relationships in forms

When using a relationship in a form, you need to select a list in the target entity to use to get a list of items in Process Experience. Following are a few suggestions about how to define and manage such lists.

- Instead of selecting a list that is used in the Lists panel in Process Experience, it is best practice to create separate "internal" lists to find related items in a form and to specifically include only the properties that are required when selecting an item from a form. For example, the Vendor list that is displayed in the Lists panel may include properties that are not needed for a user to select a related vendor in an order form or it may not include properties that are needed.
- To simplify working with the application, use a naming convention to differentiate the "internal" lists from the lists to be used by Process Experience users. For example, preface the name with INT.
- Clear the **visible to users** option on your internal lists.

Initiating an action when a property changes

In some cases, you will want to make something happen when a property changes. For example, if an application includes a Case entity with an Assignee property you may want to send an e-mail notification to the assignee when the Assignee property is entered.

To do this you'll need an extra property in the Case entity. For example, it might be called PreviousAssignee.

To create a process rule:

- Start a BPM Process if (item.Properties.PreviousAssignee == null || item.Properties.PreviousAssignee != item.Properties.Assignee)

The BPM Process should have following activities in the order shown:

1. Read ticket.
2. Send an e-mail to Assignee notifying about the assignment.
3. If PreviousAssignee != null, send e-mail to PreviousAssignee notifying that an item is un-assigned from PreviousAssignee.
4. Update PreviousAssignee = Assignee.

Validating e-mail address properties

It is not uncommon to use a text property to hold e-mail addresses. When you do so you should remember to include validation to ensure users enter valid e-mail addresses. Such validation is done using rules that have the Error action. The best way to do this is with a regular expression. You need to use Advanced mode in the rule editor to enter the rule condition:

```
! item.Properties.EMail.matches("^([a-zA-Z0-9_\\-\\.]+)@([a-zA-Z0-9_\\-\\.]+)\\.([a-zA-Z]{2,5})$")
```

Replace the name of the EMail property shown in the example with the name of your property.

Note: The ! (exclamation point) operator causes the rule to fire if the value of EMail does not match the regular expression.

Validating a date

Many times an application requires rules that contain the now or today keywords to validate that a date (such as a DueDate) is equal to or greater than the date when the item was created. Typically, this rule is written as follows:

If DueDate < today then Show error

The rule is reevaluated each time the item is opened. This works well on the Create form but could potentially cause problems when the item is opened and modified at a future date. Therefore, the rule could be valid today but it may not be valid tomorrow. The solution is to add another date property (such as DateCreated) that is set to the current date on creation. This property should not be included on a form to prevent a user from modifying it. The rule can then be written as follows:

```
if ((DateCreated is Empty && DueDate < today) OR (DueDate < DateCreated)) then Show error
```