



AppWorks™ Platform Low-Code Design Guide

Release 16.6

This documentation has been created for software version 16.6.

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 <http://www.opentext.com>

Copyright © 2019 Open Text. All Rights Reserved.

Trademarks owned by Open Text.

One or more patents may cover this product. For more information, please visit
<https://www.opentext.com/patents>

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: 4/30/2019

Table of Contents

Chapter 1 Welcome to Help for the AppWorks Platform Low-Code Designer	15
Application development overview	16
Adding functionality to the application	17
Publishing and testing the application	17
Planning the application	18
Running the application	19
Running an application on a mobile device	21
Documenting the application	24
Application development environment	24
Setting up the development environment	25
Working as a team	26
Choosing a development setup option	27
Creating a workspace and project	31
Organizing your project	34
Basic operations	35
Publishing a project	35
Testing a project	36
Packaging and deploying an application	37
Chapter 2 Quick start	39
Upgrading the Quick Start application	40
Before you begin	41
Create entities	44
Change security settings	46
Define relationships	47
Add properties	50

Create lists	54
Create forms	56
Test your application	61
Clearing the cache and refreshing the display	62
Categorizing your lists	62
Create rules	66
Create item layouts	70
Create a home page	74
Copying a building block	78
Take the next step	79
Chapter 3 Building the foundation	81
Entities	81
Creating entities	82
Entity or property	83
Adding building blocks to entities	85
Naming building blocks	88
Using toolbars and icons	89
Importing entities from database tables	91
Translating entities	95
Adding properties	96
Standard property attributes	96
Additional attributes	99
Type-specific attributes	100
Specifying an image for a property	105
Creating a business process for a dynamic enumeration	105
Null Values	111
Validation	111
Identity	112
Business ID	112
Entity status	113
Adding relationships	113
Types of relationships	113
Relationship directions	114

Multiplicity	115
Relationship possibilities	115
Relationship names	117
Defining relationships	117
Deleting relationships	119
Tracking history	119
Showing relationship actions in history	121
Tracking changes	121
Configuring security	122
Roles-based access control	123
Instance-specific entity permissions	123
Instance roles	124
Interaction between the Sharing and Security building block	124
Runtime share panel	124
Permission assignment cascading over entity relations	125
Conditional permissions	126
Default closed security configuration	126
Entity permissions	126
Building block permissions	127
Assignee task permissions	130
Business workspace task permissions	131
Content task permissions	131
Discussion task permission	133
File task permissions	133
Lifecycle task permissions	134
Activity flow permissions	135
Setting permissions	135
Granting conditional permissions	137
Chapter 4 Defining the process flow	147
Lifecycle	147
Activity Flow	147
Business Process Model (BPM)	148
Configuring a lifecycle	149

Case-centric solutions	149
Process-centric solutions	149
States	150
Transitions	151
Activities	154
Lifecycle examples	163
Adding a lifecycle	164
Creating a Lifecycle model	167
Configuring the task entity	176
Configuring expressions	181
Validating and building the model	182
Using My Inbox in the application	182
Using the Applicability Service in a Lifecycle model	184
Adding an Activity flow	193
Adding an Activity flow building block	194
Automatically initiating activity flows based on conditions	195
Integrating BPM processes and entities	196
Process overview	198
Creating the BPM process model	198
Creating the rule	200
Adding web services to the entity	201
Adding web services to the process	202
Creating an input message	203
Chapter 5 Adding business logic	205
Adding rules	205
Events that can trigger a rule	206
Advanced configurations	212
Example of configuring a rule that defines an action button	214
Availability of action buttons	214
Using web services	215
Web services on entities	219
Testing and deploying entity-based web service operations	226
Web services on relationships	226

Configuring a Find web service operation	236
Web services on other building blocks	239
Entity web service details	240
SOAP web services on the application	245
Request	254
Response	254
Request	255
Response	255
Setting deadlines	257
Adding an assignee	260
Chapter 6 Designing the presentation	265
Adding item layouts	265
Creating items using an item layout	266
Layout panels	266
Adding panels to a layout	277
Defining a custom icon for a panel header	280
Including multiple forms in a layout	280
Adding forms	281
Creating a form	282
Printing forms in the application	283
Adding properties to a form	284
Adding relationships to a form	288
Adding containers to a form	302
Adding tab containers to a form	303
Adding hyperlinks to a form	304
Adding actions to a form	305
Adding images and color to a form	306
Adding text and lines to a form	308
Designing responsive forms	309
Disabling or hiding information on a form	312
Nesting forms	313
Positioning components on a form	316
Keyboard shortcuts for forms	317

Tracking form progress	318
Drop list series example	319
Adding lists	324
Designing a list	326
Filter operators	328
Performance considerations	329
Defining lists on a tasks child entity	331
Working with tasks in application lists	333
Filtering list results in the application	341
Categorizing lists in the application	342
Adding a title	343
Creating an action bar	344
Grouping action buttons	345
Renaming action buttons and drop list labels	347
How assignment action bars behave in the application	347
Adding a mobile app	347
Configuring AppWorks Gateway per instance	348
Adding a Mobile App building block	349
Mobile App deployment messages	351
Mobile App panel support	351
Adding home pages	351
Home page example	353
Creating a theme	355
Removing a custom theme	356
Chapter 7 Managing content	357
Adding files	358
Uploading and downloading files in the application	359
Adding content	360
Working with content in the application	363
Adding a business workspace	365
Using a business workspace in the application	367
Chapter 8 Facilitating collaboration	369
Adding a message board	369

Providing email functionality	370
Creating an email template	373
Adding a business ID to an email template	374
Linking a business ID	374
Using Inbox lists	375
Configuring the E-mail connector	375
Creating a database configuration	376
Configuring the E-mail connector	377
Invoking the SetProfile Web service	386
Chapter 9 Sharing and re-using applications	389
Customizing an application	389
Configuring the custom application	390
Upgrading the base application	390
Updating security	391
Generating an application package for customization	391
Customizing platform packages	391
Importing entities from other applications	391
Managing application packages	393
Testing an application that re-uses entities from other applications	396
Using subtyping for advanced modeling	396
Replacing building blocks	398
Replacing forms	399
Replacing email	399
Conflict detection	399
Considerations for using subtyping	400
Defining security for a customization or subtype	400
Support for large tables in MS SQL	401
Chapter 10 Using the Identity Package	403
Before you begin	404
Using identity-related information in your applications	404
Entities provided in the Identity Package	404
Entities	405
Address	406

Assignment	407
Country	408
County	409
EmailAddress	410
EmailAddressType	410
EmergencyContact	411
EmergencyContactRelationship	412
Enterprise	412
Genders	413
Group	413
Identity	415
Language	416
NationalId	417
Organizational unit	418
Person	421
Phone Number	427
PhoneNumberType	428
Position	428
Role	429
SocialAccount	431
SocialMedia	431
State	432
Title	433
User	433
VisaType	435
Worklist	436
Identity domain model	438
Overview diagram	438
Detailed diagram of Person entity	439
Identity roles	439
Using OTDS with the Identity Package	439
Information pushed from OTDS to AppWorks Platform	439
Managing information pushed to OTDS	440

Mapping of OTDS identities to Identity entities	440
User	441
Group	442
Customizing the Identity package	442
Chapter 11 Using expressions	443
Selecting a property	443
Creating a basic expression	443
Creating an advanced expression	444
Using system level properties in expressions	444
Using dates in expressions	445
Using durations to specify relative dates and times	445
Using Enumerated values in expressions	446
Expression language	446
Decimal values	447
Special characters in string constants	448
Null values	448
Built-in functions	449
average and averageN	449
countTrue and countTrueN	449
max and maxN	449
median and medianN	450
min and minN	450
mode and modeN	450
optional	450
random	451
sum and sumN	451
round	451
floor	452
ceil	452
Advanced functions	452
Expressions on task states	453
Operator reference	454
Date methods	458

Date methods	460
Examples	461
String methods	461
Arrays and aggregation	463
Array processing	464
Arrays and basic operators	465
Array object types	465
Aggregation operators	466
Handling uncertainty	466
The aggregation operators	467
Chapter 12 Accessing information in external systems	469
Custom EIS connectors	469
Building a custom EIS connector	469
Defining the EIS connector	470
Implementing the generated Java classes	471
Implementing the Definition Provider class	471
Implementing the Repository Provider class	481
Packaging the application	498
Using EIS entities in an application	498
Deploying the EIS connector package	498
Configuring the connection	498
Creating an EIS repository reference and importing entities	499
Building the application	500
Deploying and maintaining the application	501
Deploying the application package	501
Maintaining the connection to the repository	501
Changing the EIS connector	502
Predefined EIS connectors	503
Creating a connection to an internal system	504
Creating an MBPM connection	506
Creating a Case360 connection	511
Chapter 13 Including advanced features	515
Designing iHub reports on entity data	515

Displaying reports in the application	517
Publishing reports to other iHub environments	518
Integrating Capture Center	519
Design considerations	520
Configuring Extended ECM for AppWorks Platform	521
Configuring a business workspace in AppWorks Platform	521
Business Workspace Configuration Report	524
Configuring Content Server to work with business attachments in AppWorks Platform	526
Configuring a cross application business workspace	526
Configuring a link to an existing business workspace	528
Configuring metadata for Documentum	530
Configuring multi-tenancy support	531
Creating an organization	532
Configuring an entity runtime database per organization	535
Deploying the application	537
Chapter 14 Maintaining the application	541
Managing solutions	541
Defining security for a solution	541
Defining security for Inbox Task Management	543
Defining security for the Identity package	544
Defining solution variables	544
Editing a solution variable for tenants	545
Changing the availability of a solution	545
Displaying a solution's errors and warnings	546
Configuring security for layouts	546
Deleting a solution	546
Deleting an entity	547
Publishing a solution	547
Deleting an EIS entity	547
Deleting history logs	548
Applying missing DDL statements	548
Selecting a database configuration	549
Specifying the email configuration	549

Editing email configuration for tenants	550
Managing identities	550
Contacts	552
Countries	554
Email address type	555
Emergency contact relationship	555
Enterprises	556
Genders	558
Groups	559
Language	559
Organizational units	560
Phone number type	562
Roles	563
Social media	563
Title	564
Users	564
Visa types	565
Worklists	566
Chapter 15 Tips and tricks	569
Developing lists of master data	569
Creating lists to use with relationships in forms	570
Initiating an action when a property changes	570
Validating email address properties	571
Validating a date	571
Chapter 16 Changing the data type of a primary key column of an imported database table	573
Impact of changing the datatype of a primary key column	573
Enabling updating of the data type of a primary key of an imported entity	574
Chapter 17 Data structure	575
Tables	575

Chapter 1

Welcome to Help for the AppWorks Platform Low-Code Designer

AppWorks Platform provides entity modeling techniques and low-code development capabilities that enable you to quickly describe the structure of business objects using a compositional approach to application development. Instead of programming certain pieces of functionality repeatedly, you define the capabilities of an entity (business object) by populating it with pre-defined functional modules called building blocks and then configuring them to suit your needs.

Entity modeling enables the following types of users to quickly and efficiently create an application that solves a business problem:

- A "citizen" programmer who understands the application requirements but who does not necessarily have programming skills.
- A software developer with programming skills.

Using conventional programming techniques, a subject matter expert such as a product manager or salesperson typically tries to communicate the business requirements to a programmer. The programmer, who may not have an in depth understanding of the business, writes the code and designs the user interface. Typically, this results in many, many iterations to review and make changes.

Using low-code development capabilities, the subject matter expert can immediately begin developing the application based on a knowledge of what it needs to accomplish. Nothing gets lost in the translation and development costs are significantly reduced. Additional cost savings result from being able to use an existing application as the basis for a new application.

Each type of user can work independently or they can both work on the same project to minimize the review/make changes cycle. For example, instead of communicating comments in person or in writing, the reviewer can work directly in the project to make the necessary changes.

A few of the advanced entity modeling functions (such as Web Services) may require some programming knowledge.

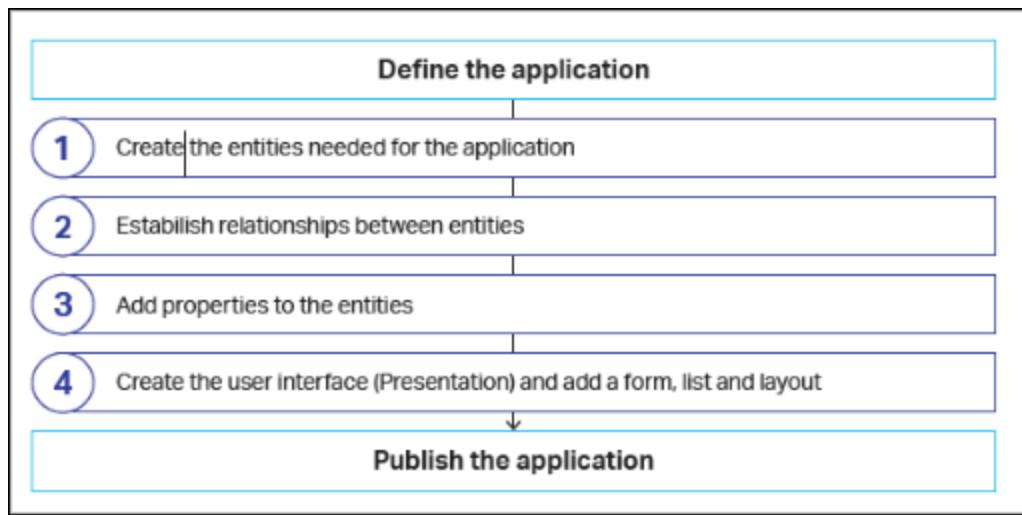
Note: In addition to entity modeling and low-code, AppWorks Platform also provides capabilities for building applications that are not entity based. These capabilities are documented in the *AppWorks Platform Advanced Development* documentation.

The [AppWorks Developer community](#) provides additional resources that you may find helpful.

Application development overview

Before you begin building an application, you need to plan what you want to build. Think about the problems it needs to solve, the information you need to collect and act on, the business processes you can automate, and how users will work with the application and what they need to see. Thinking in this way will help you build a solid information-based application domain model.

After you [create the workspace and project](#), you can create an application in four basic steps. The following diagram shows the high-level activities involved when using low-code capabilities to create an application. Once the application begins to take shape, you can add more functionality based on the requirements of the organization for which it is designed.



Step 1: Create entities - You begin creating an application by building entities. Think of the entities as the information you want to capture in the application. For example, an automobile claims application might have Claim, Policy, Vehicle, and Policy Holder entities. A healthcare application might have Physician, Patient, Allergy, and Medication entities. An HR staffing application might have Job and Candidate entities. See [Creating entities](#).

Step 2: Establish the relationships between entities - Not all entities need to be related, but when building an application, usually one or more of the entities are related in some way. So, after creating the entities you want in your application, think about how they are related. For example, in an automobile claims application, when a claim is created, the user needs to be able to select the policy holder to create the claim. Therefore, the Claim entity must have a relationship to Policy Holder. See [Adding relationships](#).

Step 3: Add properties - After you define entities and establish relationships, you can add properties to each entity. Properties in a Claim entity in the automobile claims application might include Date of Accident, Accident Location, and Driver. See [Adding properties](#).

Step 4: Create the user interface - Creating the user interface involves steps that bring the application to life for users. Typically, to begin, at least one of the following is added:

- Form - Drag and drop properties to design the form that users will use to create items in the application. For example, the form for creating a Claim might include Date of Accident, Accident Location, and Driver. See [Adding forms](#).
- Layout - Design the layout with the panels users will see when they open an item. Often layouts display a form, an action toolbar, a breadcrumb panel, and more (based upon the building blocks used in the entity). See [Adding item layouts](#).
- List - Add the properties to display to users in a list of items that were created. The properties selected for a list are shown as column headings. For example, an All Jobs list in an HR staffing application includes a list of job openings. This list might also include columns such as Job Title, Summary, Date Opened, and Hiring Manager. See [Adding lists](#).

Adding functionality to the application

At this point, the way you build the application depends on your business needs. Typically, you will not use all building blocks in all entities. The following steps are optional and you can perform them in any order that you prefer.

1. Create home page layouts that serve as landing pages for different types of users or roles.
2. Add rules that enable certain processes or actions to be performed when specific events occur.
3. Create a BPM process and prepare for integration (via a lifecycle or web services).
4. Create an entity lifecycle or activity flow to monitor how an item progresses through a business process.
5. Create action bars to specify the options that will be available to users.
6. Enable users to collaborate by attaching content to an item or by using email.
7. Add history to maintain an audit trail of changes.
8. Enable a mobile app based on entities to generate the mobile application in AppWorks.
9. Provide different levels of security access to different groups of users, so that users can perform only the operations that are granted to them on an item.
10. Identify optional integration points such as OpenText Capture Center, iHub, or other external systems.

Publishing and testing the application

As you add functionality to your application, you need to [publish](#) and [test](#) it frequently.

When all testing is complete, you can [package and deploy](#) it for production.

Planning the application

To ensure that your development proceeds efficiently and effectively, it is critical to do some advance planning.

- Define the objectives of your application. For example, a large HR application might need to track resumes, interview results, onboarding, employee records, and much more. A more simple HR application might just need to track vacation requests.
- Review the description of each building block so that you will have a rough idea of the available capabilities. See [Adding building blocks to entities](#) for a summary and click the links to more advanced topics if you want to know more.
- 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 specific entity or whether it will be part of a master list that will be related to multiple business objects (entities). See [Entity or property](#).
- Create a diagram of the entities and properties that you need and use lines and arrows to show the relationships between them.
- Consider whether you will 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, create multiple entities, each with its own Lifecycle, and relate them if necessary.
- Develop a naming convention for your entities and building blocks. This will help to keep your project organized and will make it easy to find what you need as your application grows over time. Creating guidelines for naming will also save you from having to guess about how you named a similar component. See [Naming building blocks](#)
- Think about the intended audience for your application. Will you need separate home pages for different roles or different departments?
- Keep in mind that your application will grow over time. It is best to start simply, test what you have done, and then add more capabilities.
- Compile a distribution list that identifies the users who will review your application during the development cycle and establish procedures about how they will submit their input.
- Develop a plan for creating user documentation. Usually, it is best to write a rough draft during development and then fine-tune it when the application is complete. See [Documenting the application](#).
- Decide how you will set up the environment where you will work and perform a few initial procedures. See [Setting up the development environment](#)

After you complete these planning activities, it is recommended that you proceed through the [Quick start](#) tutorial exercises to create a simple Order Management application. This will give you a familiarity with the basic steps involved in creating an application.

Running the application

AppWorks Platform provides a runtime environment where authorized users work with the applications that you create. You customize the user interface for the application by adding functional modules called building blocks and configuring them based on your organization's requirements. For example, for a real estate application, users might need to create items such as appraisal reports, viewings, and listings. You can 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.

By default, your application runs using the default home page, although you can create multiple customized home pages so that users can switch between different home pages depending on what they need to do. Panels are the primary components of a home page. Each panel provides specific functionality. For example, a panel might display a list of work items awaiting a user's attention or it might show a map indicating the location of an accident.

Home pages

A home page serves as a landing page for application users. You typically configure different home pages for different types of users or for different departments in an organization. For example, users who process claims may see a list of tasks awaiting their attention while managers may see summary reports that contain statistics about the number of claims processed, the users who completed them, and so forth. Some home pages enable users to create items such as claims, orders, and so forth. You control security to home pages using security settings. A selection menu is provided for users who have access to multiple home pages.

Each home page comprises one or more pre-defined panels. Many types of panels are available for adding to a home page. See [Layout panels](#). The default home page includes the following panels:

- The Lists panel displays the lists that the user who is signed on has permission to use. The user can reorganize the set of lists, create new sections, drag lists between sections, hide lists, and rename lists.
- The Results panel displays a list of items. You can choose to have the panel display the results of a specified list or to respond to the selection of a Lists panel elsewhere in the layout.
- The Preview panel enables application users to preview an item selected from the Results panel. The Preview panel can display many items at the same time, organizing them using tabs.

The header area at the top of the home page contains the following information:

- The user's display name.

- A list of the home pages to which the user has access. The first time a user opens the application, the default home page is displayed. If the user selects a different home page, it is remembered as the user's new default home page.
- Depending on how you configure a home page, it may or may not provide a Create menu that enables users to create items.
- The status of Auto Save, a feature that automatically saves every change a user makes. The status indicator responds to the screen size and is shown either as text or an icon. For example, on a full-size display it is shown as text. On a mobile device it is shown as an icon.

Message	Description
Saving	Your changes are being saved.
All changes saved	Your changes have been saved. The message disappears after a few seconds.
Save Error	<p>One of the following errors has occurred:</p> <ul style="list-style-type: none">■ Edit conflict - The Resolve Conflicts dialog box opens showing the entered value and the conflicting value. Select Keep all Mine or Keep all Conflicting and then click OK.■ Incorrect data type or rule error - The incorrect value is highlighted and a message describes the error. Type a valid value.■ Lost connection - Restore the connection. <p>After the issue is resolved, another save attempt is made.</p>

How users create items

When a user creates an item using the Create menu on a home page, a list of all items that are available to be created is displayed. A list of recently created items is also available.

Users can also create items from a list or from a Create panel. When using the Create panel, a list of recent items is not available and items that are created within the Create panel are not shown in the list of recent items when you create an item using the Create menu (+) on a home page.

After users enter the information required to create an item, they have the following options:

- **Create** creates the item and closes the dialog box.
- **Create and open** creates and opens the item.
- **Save and create another** creates the item and re-displays the Create dialog box.

Running an application on a mobile device

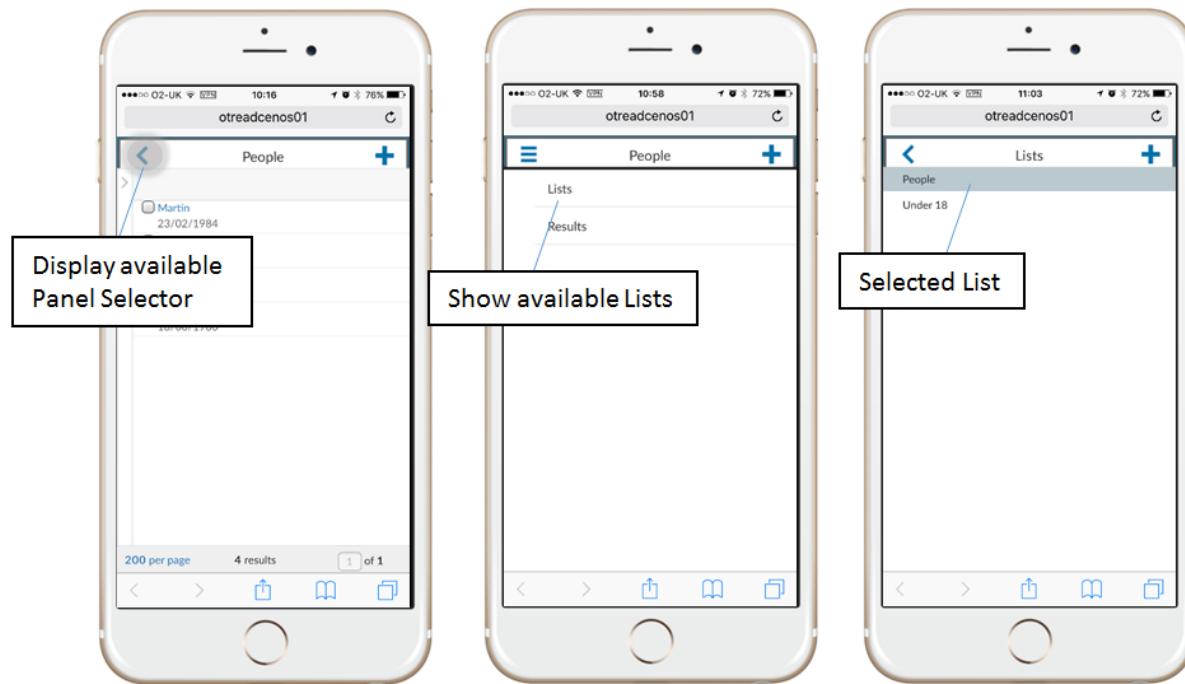
There are some differences in the user experience when users access the application using a URL within a mobile web browser. The user has the same content available as when using a desktop browser but the experience is tailored to a mobile device. The differences are described in this topic.

When users access the application URL on a mobile device, the layout panels are organized in a way that is optimized for simplicity and ease of use.

- Form components are displayed in a single “stack-view” column.
- Columns in lists are displayed as rows from left to right and top to bottom – “Paragraph” display.
- Each panel occupies as much height as needed, with vertical scrolling for pages accessing additional content.

Layout and navigation

The first time users access the application, the Home Page is displayed and the Results panel is presented. To populate the Results panel, users need to navigate to the Lists panel and select a list.



Users can return to the Results panel by going back to the panel selector and selecting Results.

To return to the Home Page at any time, users can go to the panel selector and click the Actions menu () to display all available home page layouts. Tapping the required home page displays the Results panel of the most recently selected list, where applicable.

Creating an item

Users can create a new item by clicking the create icon on the header bar. The Quick create page is displayed with all available entity creation options (based on your permissions). Tapping an option displays the Create form.

Panels in mobile

The following sections describe each panel and how it is used on a mobile device.

Results

The Results panel can either be driven by an associated List panel, or can automatically run a predefined List for the entity.

The Results panel in mobile presents the data in paragraph mode. This enables more data to be available on the smaller screen size. Each column of data is in rows for each instance. The column headers are not displayed to preserve space.

Users can change the paging that you specify by tapping the **X per page** in the footer. The number of results displayed on the page is shown in the center of the footer. Users can manually modify the paging by changing the page number in the footer or they can choose the next or previous page by tapping the arrow.

Users can filter the results by tapping the left portion of the screen. The filter options can be applied to each column of data to narrow the results list. Applied filters are displayed at the top of the Results and can be cleared by tapping the X.

To perform actions on an item, users tap the check box in the row required for action. Actions are displayed in the action bar below the header bar and are collapsed into an ellipsis if the text is too wide to display in the available space. Tapping the ellipsis provides access to the additional actions.

Tapping the blue text in the Results opens the item. Optionally, open is also available from the action bar. To return to the Results, users access the hamburger menu and tap the desired Home Page.

Form

A form is responsive and adapts to enable usage within a mobile device. Fields appear in "paragraph" form. The Form panel displays the data in a single column. This stacked view enables each element to present the data in an easy to read way.

Containers are shown as separate panels that users swipe to.

Note: Users cannot open an associated item because there would be no way to navigate back to the original item.

Contents

The Contents panel operates in the same way as in the desktop application. However, the native device options are launched when uploading a file.

Discussions

The Discussions panel operates in the same way as in the desktop application. The Discussions panel is built to be fully responsive so that the content within the panel adapts automatically to the mobile device.

Web Content

The Web Content panel presents a web page as specified in the design.

Note: Some external websites prevent the display of their web pages.

Tasks

The Tasks panel enables the progress of the Lifecycle building block tasks. The tasks defined for the Lifecycle are represented in a paragraph list. Users can perform task actions by checking the row and selecting the action from the action bar menu. Tapping the row also displays an ellipsis, with a further tap displaying the available actions for the task.

Each task row contains the option to select the current status of the task (represented as an icon), the task title, the assignee, and the due date.

Within the action bar of the Tasks panel, the filter icon opens the **refine by** menu. This menu provides options to narrow the task results based on status, assignment, and due date.

Users can remove the filter criteria by either tapping the checked items in the **refine by** menu, by tapping the x within the tags on the filter bar, or by tapping **Clear all**.

The plus icon enables users to create manual follow up tasks for the current Lifecycle state.

Actions in mobile

This section describes actions that are displayed differently within a mobile device.

History

The history display is presented in paragraph mode. This enables easy reading on a mobile device.

File - Upload

Using the Upload action to select a file uses the native file upload for the device:

- **iOS** - The action sheet displays options to Take Photo or Video, Photo Library, iCloud Drive, and any other document provider defined on the local device.
- **Android** - This is determined by the phone brand and version of android but the option to take a new photo or upload an existing photo is the minimum provided from the native controls.

File - Download

The Download action is available only on Android devices. iOS devices do not allow access to the file system from the browser.

Lifecycle building block – State transitions

User state transitions are displayed in the action bar when available. These options move the lifecycle to the next state or complete the lifecycle.

Add Tasks

The Add Tasks action presents a dialog box to add manual tasks defined for the building block.

Documenting the application

Since each application is highly customized, it is strongly recommended that you provide documentation for the users who will work with it. The content of the documentation depends on the building blocks you have included and how you have configured them.

Writing the documentation is typically done as you add capabilities and test them in the application. You can use the tool of your choice. One simple way is to create a Word document, save it as a PDF file, and then place a link to it on the application home page. It is best to provide the help in a searchable format.

Many sections of this document provide detailed information about how certain features are used in the application. You may be able to use this information as a starting point for creating your own customized documentation.

Application development environment

You develop your application in the AppWorks Platform Collaborative Workspace (CWS) environment. Before you can begin, you need to review some planning guidelines and perform a few tasks.

See the following topics:

- [Setting up the development environment](#) provides general planning guidelines and a few links to relevant procedures.
- [Choosing a development setup option](#) describes the development setup options and provides the advantages and disadvantages of each.
- [Creating a workspace and project](#) provides an overview of workspaces and projects and explains how to create them.
- [Working as a team](#) explains how to add your project to source control.
- [Organizing your project](#) explains how to create folders in which you can organize the various components of your project.

Setting up the development environment

The following list provides some recommendations for setting up your development environment and contains a few links to specific procedures.

Configure the AppWorks Platform development environment

Review product documentation and relevant How To articles for configuring the development environment.

Develop a working knowledge of the software needed for development on the client computer.

Software	Function
Google Chrome	Browser
Eclipse IDE for Java Developers	Java development
Eclipse SVN Subclipse plugin	Source control from within Eclipse
Visual Web Developer Express	Front-end Javascript editing and debugging
SOAPUI	To test web-services
Fiddler	To see exact traffic between browser and web gateway
IE Developer Toolbar	Check HTML in browser
Notepad++	To edit files

Initialize the SVN repository

The SVN repository that will contain the application sources is likely to contain other project related documents as well. Think about documentation, database scripts, and continuous integration scripts. See [Working as a team](#)

Define unique qualified names for all application components

To avoid name collisions with arbitrary application components of third party applications, it is recommended to carefully think about unique qualified names for all the application components that will be developed. Think, for example, about defining the Java package structure in case part of the business solution will be implemented in Java.

Define the projects within the workspace

Developing an application starts with defining the workspace structure. See [Creating a workspace and project](#).

Define the folder hierarchy for each project

The initial phase of developing the application mainly consists of setting up the project hierarchy in terms of folders, sub-folders and start points of qualified names. This part can best be done in advance by a single developer to prevent developers from blocking each

other because of documents being locked in the SVN repository. Although it is possible to change the project hierarchy when this is required, having a well-though project hierarchy from the start is much preferred.

Properly set the package properties (for example, the owner, product, version, and so forth) for each project.

See [Organizing your project](#).

Other recommendations

- To ensure that individual developers are testing their own work in progress, every developer must work in a separate organization. The System organization can best be used for system administration purposes only and not for application development.
- Configure a dedicated organization for the build master. This provides an isolated environment for building and packaging the application under development after each commit to the SVN repository.
- If no test server is available, configure a dedicated organization for functional testing. Create a single workspace in this organization that is updated with the latest sources whenever a functional test must be performed. Clean the build folder to ensure that publishing the projects to be tested builds and deploys all corresponding artifacts afresh.
- For all matters related to connectivity from the platform to other backends, take a close look at the Connector Directory.

Working as a team

In most organizations, there isn't just one developer working in a workspace, but rather, there is a development team. The platform supports various ways to configure workspaces to support development teams. The best practice is to create a separate workspace for each developer on the team, all connected to a common SVN source control repository.

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 area for each team member as shown 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**.

Choosing a development setup option

There are four development setup options, each with its advantages and disadvantages:

- [Organization per developer](#)
- [Product instance per developer](#)
- [Single organization with a workspace per \(group of\) developer\(s\)](#)
- [Single organization with a single common workspace for all developers](#)

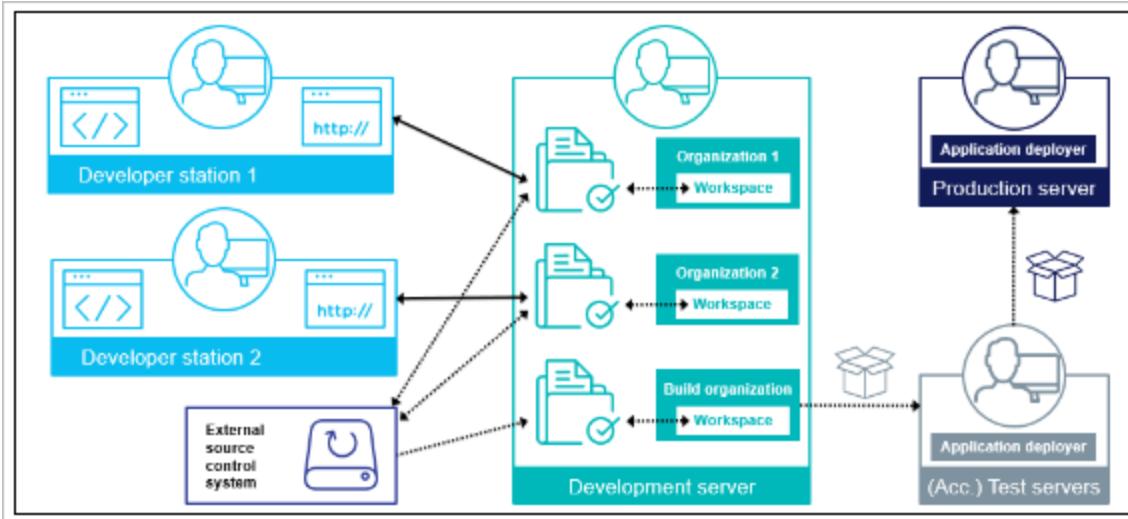
While one is recommended as a best practice, there may be situations where another is more suitable. You can view the alternatives in the following sections.

Recommendation - It is recommended that you create a separate organization per developer, which therefore implies a separate workspace for every developer. Although this increases hardware resources (especially memory), the advantages in managing each developer's code outweighs the additional overhead in maintaining this setup.

Note: When working on a file system for web content, such as JavaScript or stylesheets, a shared workspace must be used.

The following sections describe each setup option.

Organization per developer



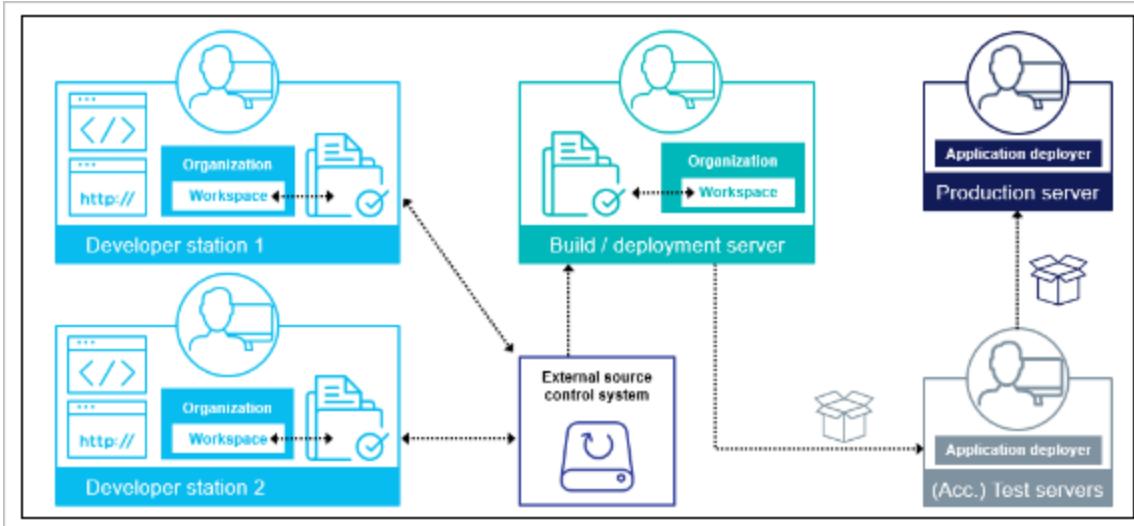
Advantages

- In most cases, full separation of sources and published content
- Check-ins are in the developer's name
- Developers can check in when their work is completed
- No interference with other developers' work (unless software that contains bugs is checked-in)
- What the developer publishes is what the developer tests - other developers cannot write over published content

Disadvantages

- Extra work is required to maintain this setup
- Every organization needs its own configuration, and likely its own application database
- More hardware resource intensive, especially memory

Product instance per developer



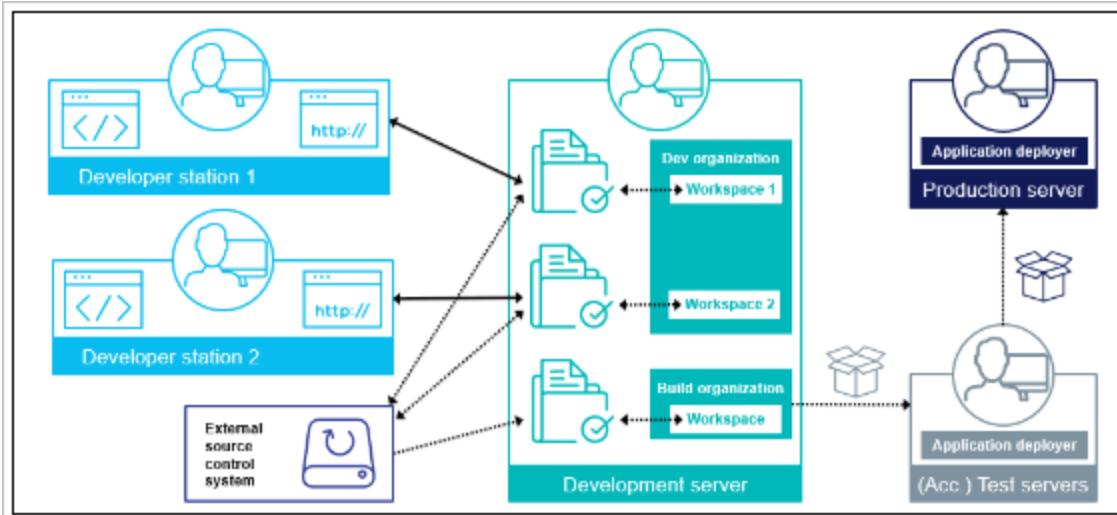
Advantages

- In most cases, full separation of sources and published content
- Check-ins are in the developer's name
- Developers can check in when their work is completed
- No interference with other developers' work (unless software that contains bugs is checked-in)
- What the developer publishes is what the developer tests - other developers cannot write over published content

Disadvantages

- Extra work is required to maintain this setup
- Every organization needs its own configuration, and likely its own application database
- More hardware resource intensive, especially memory
- Risk of incompatibility of developed artifacts if not all product instances are on the same version
- No offline development possible - a permanent connection to SVN is required

Single organization with a workspace per developer



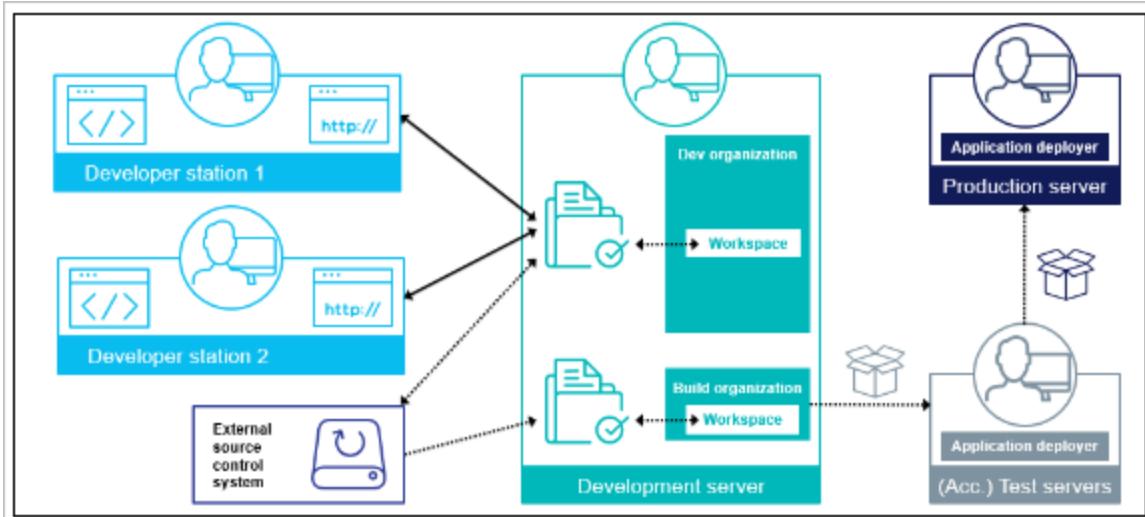
Advantages

- Easy to maintain and set up
- All organization-specific configuration needs to be done only once
- Moderate hardware resources are required
- Check-ins are in the developer's name
- Developers can check in when their work is completed

Disadvantages

- Unfinished changes might make it impossible for other developers to test their own changes
- When performing a full build and publish, a developer may overwrite the published version of other developers' work
- More time may then be spent fixing these conflicts or discovering why something doesn't behave as expected

Single organization with a common workspace for all developers



Advantages

- Easy to maintain and set up
- All organization-specific configuration needs to be done only once
- The least hardware resource intensive

Disadvantages

- Check-ins to Subversion are not in the developer's name
- Unfinished changes might make it impossible for other developers to test their own changes
- Questions around when to commit are raised: only when everyone is done, or when only one person is done - which can be complicated to sort out

Creating a workspace and project

You create an application within the AppWorks Platform Collaborative Workspace (CWS) environment. Before you can begin, you need to create your workspace and project.

Creating a workspace

A workspace is the starting point for low-code application development. A workspace contains all the projects that you work with. In most organizations, there isn't just one developer working in a workspace, but rather, there is a development team. AppWorks Platform supports various ways to configure workspaces to support development teams. The best practice is to create a separate workspace for each developer on the team, all connected to a common Subversion (SVN) source control repository. When used this way, each developer's work is isolated in a separate workspace. Developers can check in their work to make it available for sharing with other team members. See [Working as a team](#).

In addition to sharing projects, the Imported Application Packages area enables you to use applications from other developers, even though you do not have the source code for these packaged applications. These can be purchased applications or applications from other teams in the same enterprise. An example of an OpenText application that is available out of the box in the Application Packages area is the [Identity Package](#).

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:). Therefore, 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:

1. Start AppWorks 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. You can use it to control the versions of your application or when working with multiple developers to create applications. 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. CWS supports only http and https.

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

See [Working as a team](#) and the *AppWorks Platform Advanced Development* documentation for details.

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 [AppWorks Administration](#). 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 delete a workspace:

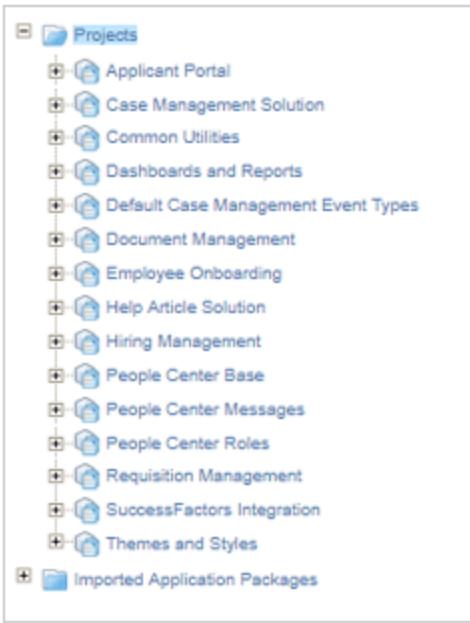
- In Organize Workspaces, select the workspace and then click  (Delete).

Creating a project

A workspace contains any number of projects. A project corresponds roughly to an application. Projects can vary from simple to complex, depending on the type of application. 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. AppWorks Platform provides options to help you organize your projects to make your applications easy to maintain.

A project contains a set of artifacts called documents. Many types of documents can be added to a project and the list continues to grow as new capabilities are added to AppWorks Platform. See [Organizing your project](#).

The following example shows an application that contains several projects.



To create a 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 Package Owner and the Product Name are properties of the application package for delivery.
- The new project is created and added to the Project List.

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. There are many ways to use folders - one approach is to create a folder for each type of model in your project. The set of folders needed for a project will vary, but it is recommended that no empty folders are created.

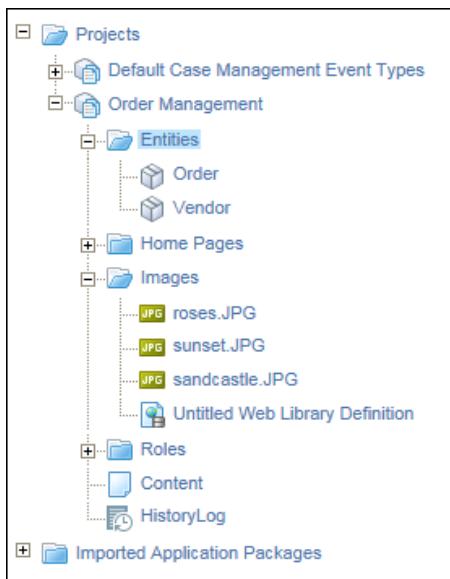
A project could have the folders as an example:

- Audit Logs
- Business Processes
- Database Metadata
- Entities
- Java
- Roles
- Web Services

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.



Basic operations

As you develop an application you need to frequently publish it to the runtime so that you can test what you've done.

After you test your application and verify that it works as you intended, you can create and download an application package, and then deploy it to your production environment.

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 the application in another so that you can easily move between the design environment and the application 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 the application. 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 the application. Therefore, if you create entities with lists, forms, layouts, and so forth and then publish the project, you will not be able to see your application unless you edit the security settings. You need to define solution security only after you publish the project for the first time. See [Defining security for 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

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 [AppWorks Administration](#), 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 the application 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 the application.

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 an AppWorks 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.

Note: Entities can only be used in a package created for Run Time. They cannot be used in a package created for Staging.

For detailed information on packaging and deploying your applications and solutions, see the *AppWorks Platform Advanced Development* documentation.

Chapter 2

Quick start

After AppWorks is installed, you are ready to experience the simplicity of building an entity-based application. 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 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 work with 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.

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 the steps exactly as written, 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](#)
- [Create item layouts](#)

- [Create a home page](#)
- [Copying a building block](#)
- [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, until you configure the user interface, there will be nothing to display in the application or 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. After you publish, press CTRL + F5 in your application to update it with your changes.

The objective of the Quick Start tutorial 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.

Upgrading the Quick Start application

Effective with AppWorks Platform 16.4, the Quick Start tutorial adds two new exercises:

- Creating an item layout
- Creating a home page layout

If you created the Order Management Quick Start application using an earlier version, you have two options for taking advantage of these enhancements. You can either recreate the application completely or you can modify your application by making the following minor changes in the Vendor entity:

- Create a text property with a display name of Zip Code and a name of ZipCode.
- Add the Zip Code property to the All Vendors list and to the Create and Default forms for the Vendor entity.
- Enter the Zip Code value for any vendors that you previously created in the application. Although the name and street address of the vendor can be fictitious, the zip code should be compatible with the city and state. You can also add a few new vendors.

See the relevant topics, which have been updated to include these instructions and new screenshots if necessary.

Before you begin

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

Your application designer or system administrator can provide the specific URLs to use for each environment.

- Collaborative Workspace (CWS) is the design time environment where you create your entity-based application. CWS is part of AppWorks Platform. During most of the Quick Start tutorial, you will be working in this environment.

`http://<hostname>/home/<org>/`

- The application runtime environment is where you test your application as a user sees it.

`http://<hostname>/home/<org>/app/start`

- AppWorks Administration is where you set security options.

`http://<hostname>/home/<org>/app/admin`

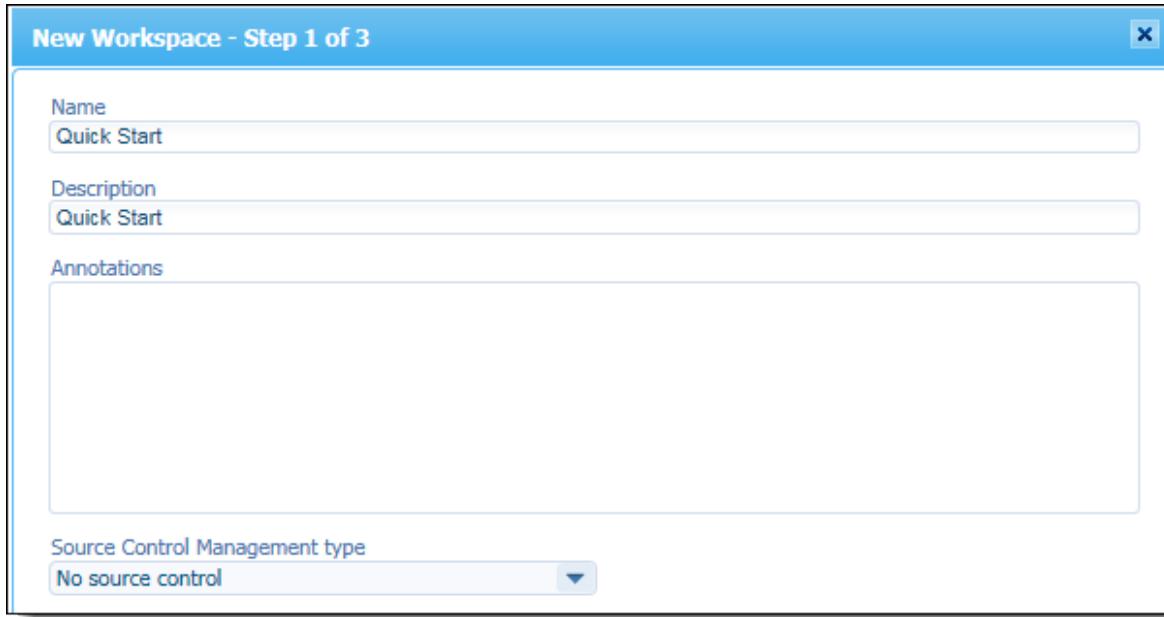
A workspace is the starting point for application development and is the place where you maintain your projects. Before you can begin building your application, you need to create a workspace and project. To create a workspace, you must have the Developer role.

Important

- OpenText recommends that you maximize each environment while working in it.
- If multiple users are running the Quick Start on the same server, include your name in the workspace name to avoid naming conflicts. For example, name the workspace **Ben Quick Start**.

To create a workspace and project in CWS:

1. In AppWorks Platform, if the list of applications are not visible, click the small arrow to pull down the list of applications.
2. 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**. Ensure to include your name if multiple users are running the Quick Start on the same server. For example, type **Ben Quick Start**.
 - The Description is automatically filled.
 - Leave Annotations blank.
 - In Source Control Management type, leave the default selection of No source control.

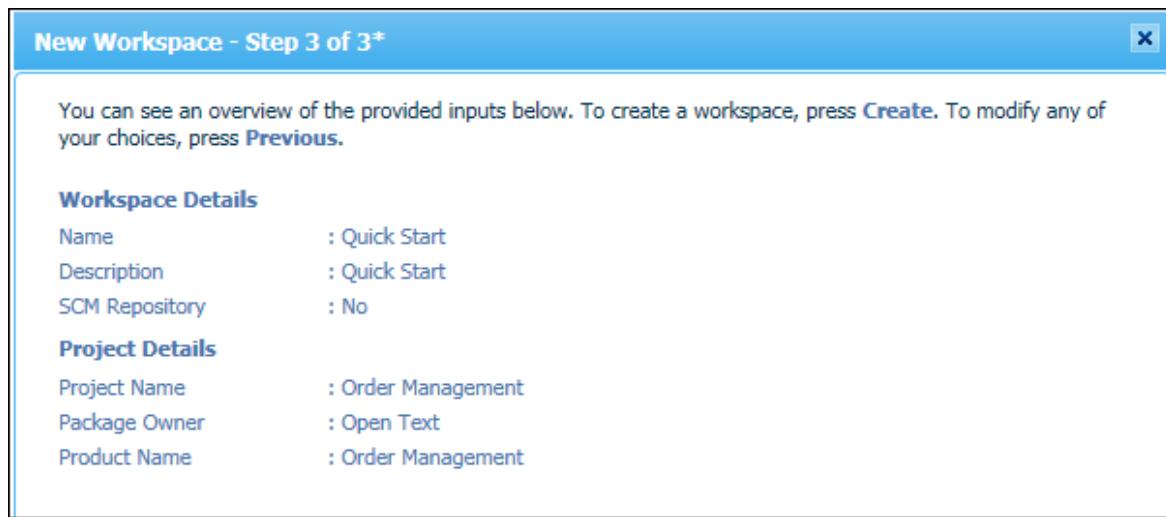


5. Click **Next**.
6. In the New Workspace - Step 2 of 3 window:
 - In Project Name, type **Order Management**. The Product Name is filled with the Project Name.
 - In Package Owner, type **Open Text**.

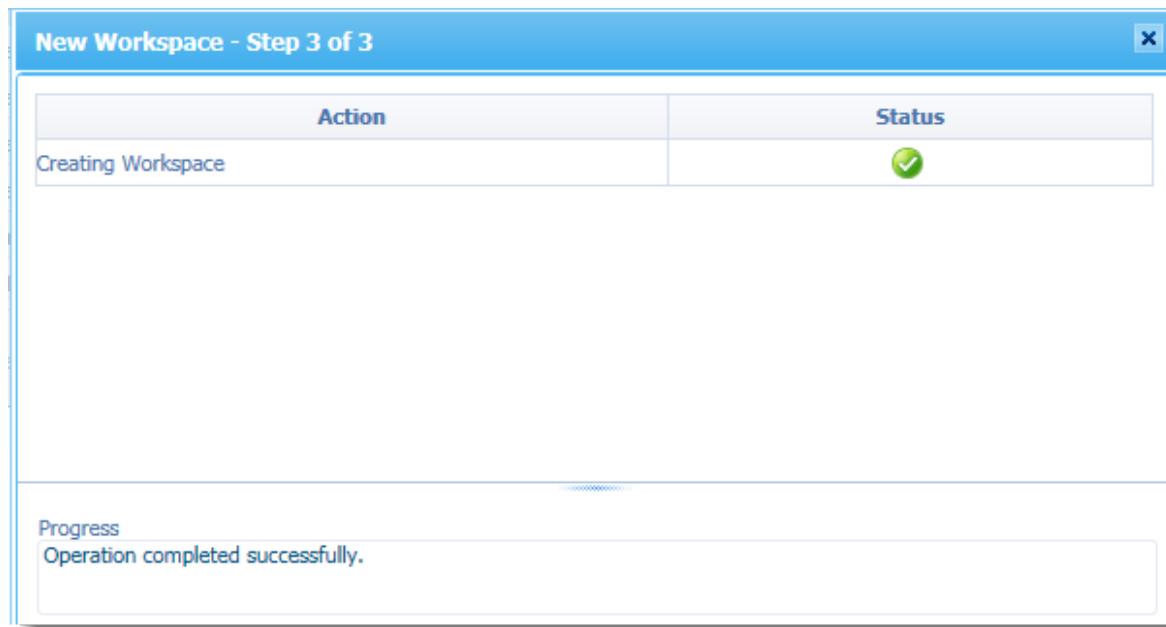
Note: If multiple users are running the Quick Start on the same server, include your name in Package Owner, for example, type **Open Text Ben**.



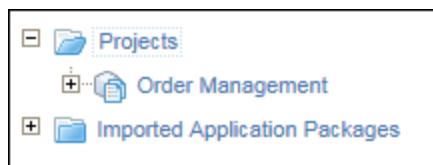
7. Click **Next**. A workspace summary is shown. If necessary, click **Previous** and make modifications.



8. Click **Create**. A message appears confirming the creation of your workspace.



9. 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

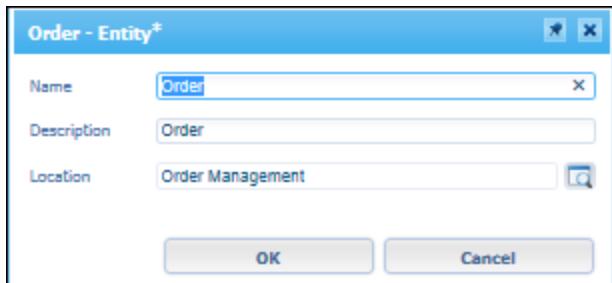
In this exercise you will create entities (business objects) called Order and Vendor. Each entity has a Display name and a Name.

- The Display name is used as an option for creating an order or a vendor in the application. For example, if the Display name is Order, the Create a new item 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 suggestions are automatically displayed. 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. The entity is created with a name of Untitled_Entity (shown in the title bar). An Identity building block is added to the entity. 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 and you can make changes if necessary. The asterisk in the title bar indicates that the entity has not been saved.



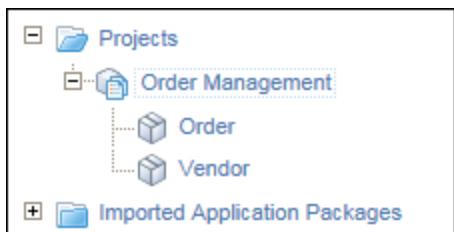
9. Click **OK**.

To create the Vendor entity:

1. In the Order - Entity window, click the arrow between (Workspace Documents) and (New).
The Quick Access Menu opens.
2. Click **Entity** or type **entity** in the search box, and then click **Entity** in the search results.
The entity is created with a name of Untitled_Entity and an Identity building block is added to the entity.
3. In the Untitled_Entity window, click .
4. In Display name, type **Vendor <your name>**.
5. In Name, type **Vendor**.
6. Leave all other options at their default settings.
7. Click (Save).
The information you entered is displayed and you can make changes if necessary. The asterisk in the title bar indicates that the entity has not been saved.
8. 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. Therefore, keep these windows minimized for convenience.



What's next?

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

Change security settings

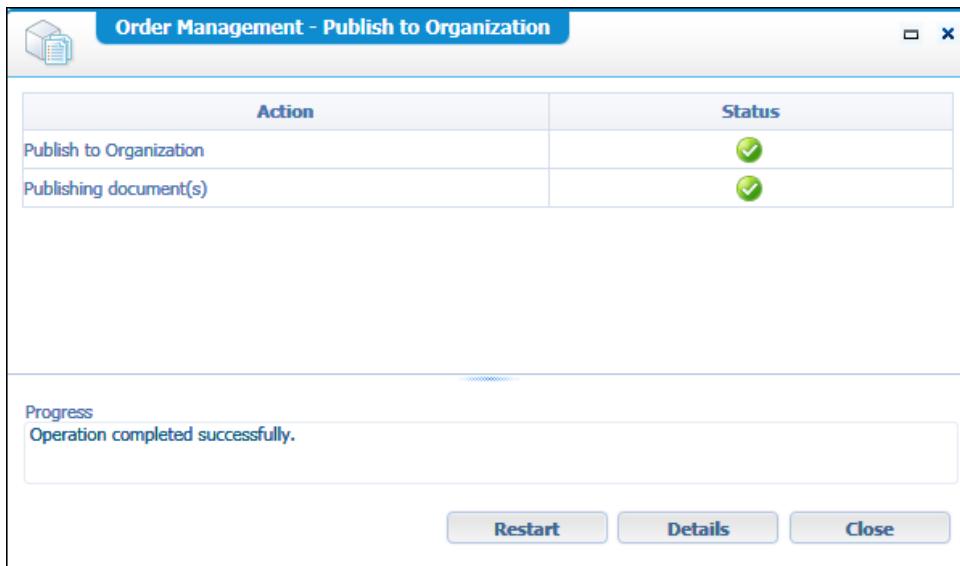
By default, applications are not visible to users until you publish your project and configure security using [AppWorks Administration](#).

Before you begin:

- Consult your system administrator to obtain the [URL](#) for running AppWorks Administration. 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**. The following dialog box opens when the operation is complete.



2. Click **Close**.

To configure the security settings:

1. Run AppWorks Administration.
2. Press F5 to refresh the display.
3. In Solutions, select your solution: **OpenText<your name>OrderManagement**.
The solution 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 then select **Use solution**.
5. In Roles, ensure 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

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

With your workspace selected in the Solutions pane, the Configurable elements and Roles panes must look like this.

The screenshot shows two panels side-by-side. The left panel, titled 'Configurable elements', contains a tree view of solution components: 'Solution status', 'Solution security' (expanded to show 'Administer solution', 'Build', 'Include', 'Manage solution', and 'Manage solution security'), and 'Use solution' (which is highlighted with a blue background). The right panel, titled 'Roles', lists various role checkboxes. Most checkboxes are unchecked except for the first three, which are checked: 'Entity Runtime Administrator of OpenText Entity Runtime', 'Entity Runtime User of OpenText Entity Runtime', and 'Entity Runtime Developer of OpenText Entity Runtime'.

Role	Status
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 checked="" 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	<input type="checkbox"/>

The solution is now available for every application user. When you develop actual applications that are put into production, you must 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 Order and Vendor entities.

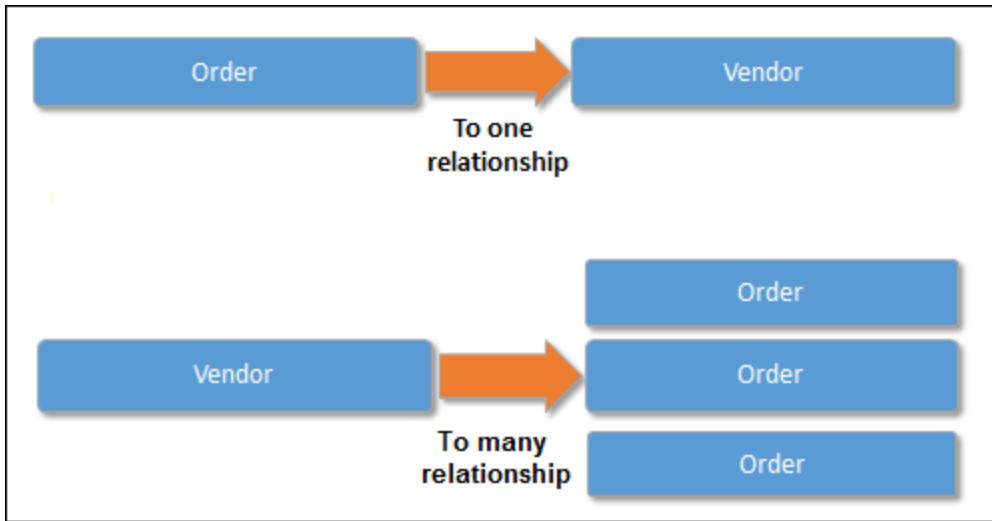
Define relationships

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 specific order or immediately see vendor details when opening the form.

For the Order Management application, create the following relationships:

- Each order has only one vendor. Therefore, create a To one relationship from Order to Vendor.
- Each vendor can have multiple orders. Therefore, 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 in one of the following ways:
 - Double-click it in Workspace Documents.
 - Click the minimized window for the entity.
2. Navigate to Available building blocks > Structure, and then click **Relationship**. The Add relationship dialog box opens.
3. In the Add relationship dialog box, define the following options for the relationship:
 - In the relationship type list, leave the default value of relates to one.
 - Click **Browse** and select **Vendor** from the Select target entity dialog box.
 - In Label, type **Vendor**. In Name, the Label value is filled by default.
 - Ensure that the **Make this relationship bidirectional** check box is clear.

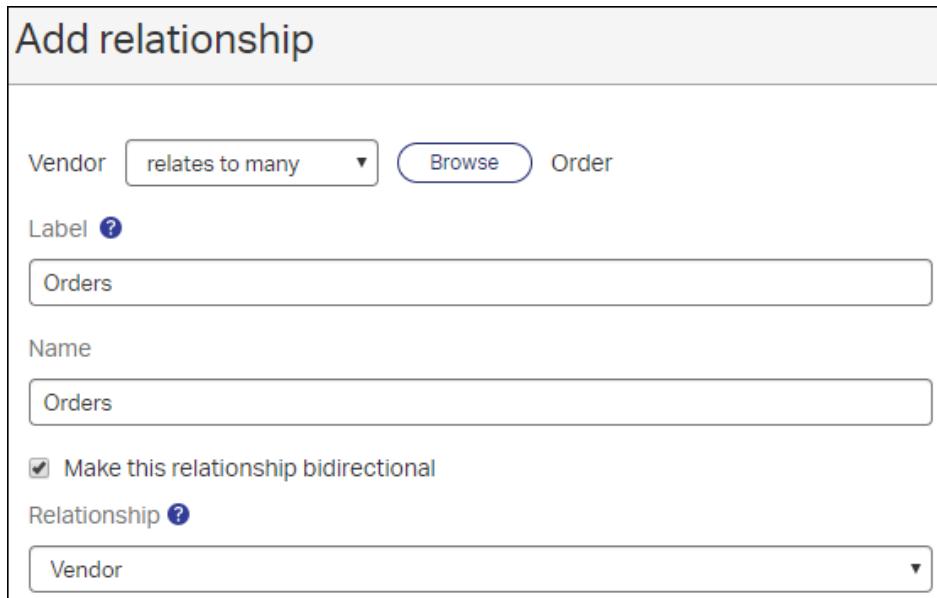
The 'Add relationship' dialog box shows the following configuration:

- Relationship Type:** Order relates to one Vendor.
- Label:** Vendor
- Name:** Vendor
- Options:** Make this relationship bidirectional (unchecked)

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

To relate vendors to orders:

1. Open the Vendor entity in one of the following ways:
 - Click the related entity Vendor hyperlink in Configuration in the relationship details pane of the Order entity.
 - Double-click it in Workspace Documents.
 - Click the minimized window for the entity.
2. Navigate to Available building blocks > Structure, and then click **Relationship**. The Add relationship dialog box opens.
3. In the Add relationship dialog box, define the following options for the relationship:
 - In the relationship type list, select **relates to many**.
 - Click **Browse** and select **Order** from the Select target entity dialog box.
 - In Label, type **Orders**. In Name, the Label value is filled by default.
 - Select the **Make this relationship bidirectional** check box.
A list displays the relationships available to make this relationship bidirectional.
 - In the Relationship list, select **Vendor**.



The screenshot shows the 'Add relationship' dialog box. At the top, it says 'Add relationship'. Below that, there's a section for 'Relationship type' where 'Vendor' is selected and 'relates to many' is chosen from a dropdown. Next to it is a 'Browse' button which has 'Order' selected. Under 'Label', the word 'Orders' is typed. Under 'Name', it also says 'Orders'. There's a checked checkbox for 'Make this relationship bidirectional'. At the bottom, under 'Relationship', 'Vendor' is selected from a dropdown.

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

Note: To check your work, open Workspace Documents, right-click the **Order Management** project, and then select **Validate**. Correct any errors that are shown and close the window after validation is successful.

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 for a user of your application to 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 the Order Management Quick Start, item is defined as a text property for simplicity.

Properties are often used in forms to display values 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. Navigate to Available building blocks > Structure, and then click **Property**.
The Add property dialog box opens.
3. In the Add property dialog box, add the following properties.
Select **Add another** for the first property, and then click **Add** after adding each property. For the last property, clear **Add another** before clicking **Add**.

Label	Name	Type
Order Number	OrderNumber	Text
Order Date	OrderDate	Date
Item	Item	Text
Quantity	Quantity	Integer
Color	Color	Enumerated text Note: In Enumeration type, leave the default value of Static.

All the properties are added to the Properties building block.

4. In the Color details pane, in Configuration, configure the **Color** property as follows.
 - In the displayed table, in the Label column for none, type **None** and click **+** (Add).
 - In the Label column of the new row, type **Red** and press Tab. In the Value column, the Label value is filled by default.
 - Click **+** (Add).
 - Continue adding the following colors: **Yellow**, **Blue**, **Green**, and **Purple**. The list must look as follows.

Label	Value	Default	Icon
None	none	<input checked="" type="radio"/>	
Red	Red	<input type="radio"/>	
Yellow	Yellow	<input type="radio"/>	
Blue	Blue	<input type="radio"/>	
Green	Green	<input type="radio"/>	
Purple	Purple	<input type="radio"/>	

Sort list by label

5. In Advanced configuration, in Length, leave the default value. (Do not change any of the other values).
6. Click **💾** (Save).

Optional. To verify your work, open Workspace Documents, right-click the **Order Management** project, and then select **Validate**. Correct any errors that are found and then close the window after validation is successful.

When you open the Order entity, the properties that you added are shown in the Properties building block in the Added building blocks pane. 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. Navigate to Available building blocks > Structure, and then click **Property**. The Add property dialog box opens.
3. In the Add property dialog box, add the following properties.
Select **Add another** for the first property, and then click **Add** after adding each property. For the last property, clear **Add another** before clicking **Add**.

Label	Name	Type
Vendor Name	VendorName	Text
Vendor ID	VendorID	Text
Street	Street	Text
City	City	Text
State	State	Text
Zip Code	ZipCode	Text
Status	Status	Enumerated text Note: In Enumeration type, leave the default value of Static.

All the properties are added to the Properties building block.

4. Configure the **Status** property in the Status details pane as follows (do not change any of the other values):
 - In the displayed table, in the Label column for none, type **None** and click **+** (Add).
 - In the Label column of the new row, type **Active** and press Tab. In the Value column, the Label value is filled by default.
 - Click **+** (Add).
 - In the Label column of the new row, type **Inactive** and press Tab. In the Value column, the Label value is filled by default.
 - In the Default column, select the row for **Active**.
 - Select **Sort list by label**.

The list must look as follows.

Status

Summary
Adds a property to an entity. Properties are of various types and can have constraints.

Configuration

Label
Status

Type
Enumerated text
Enumeration type
Static

Label	Value	Default	Icon
None	none	<input type="radio"/>	[Image]
Active	Active	<input checked="" type="radio"/>	[Image]
Inactive	Inactive	<input type="radio"/>	[Image]

Sort list by label

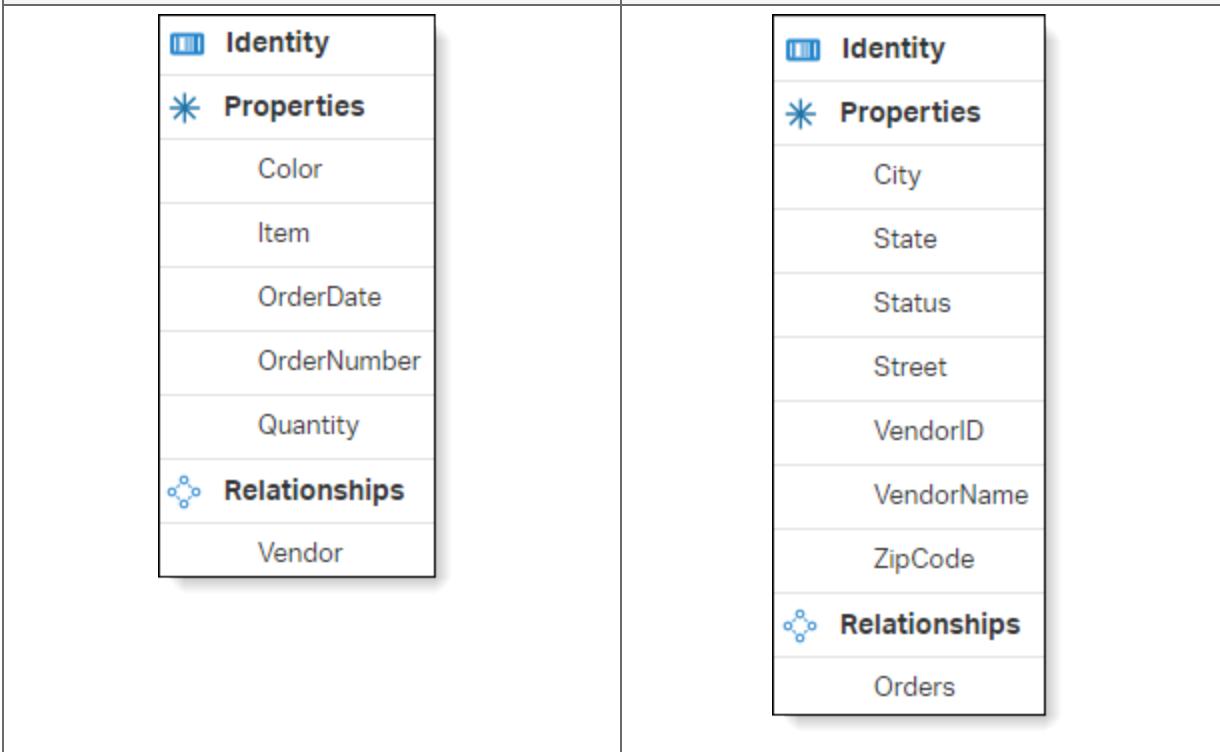
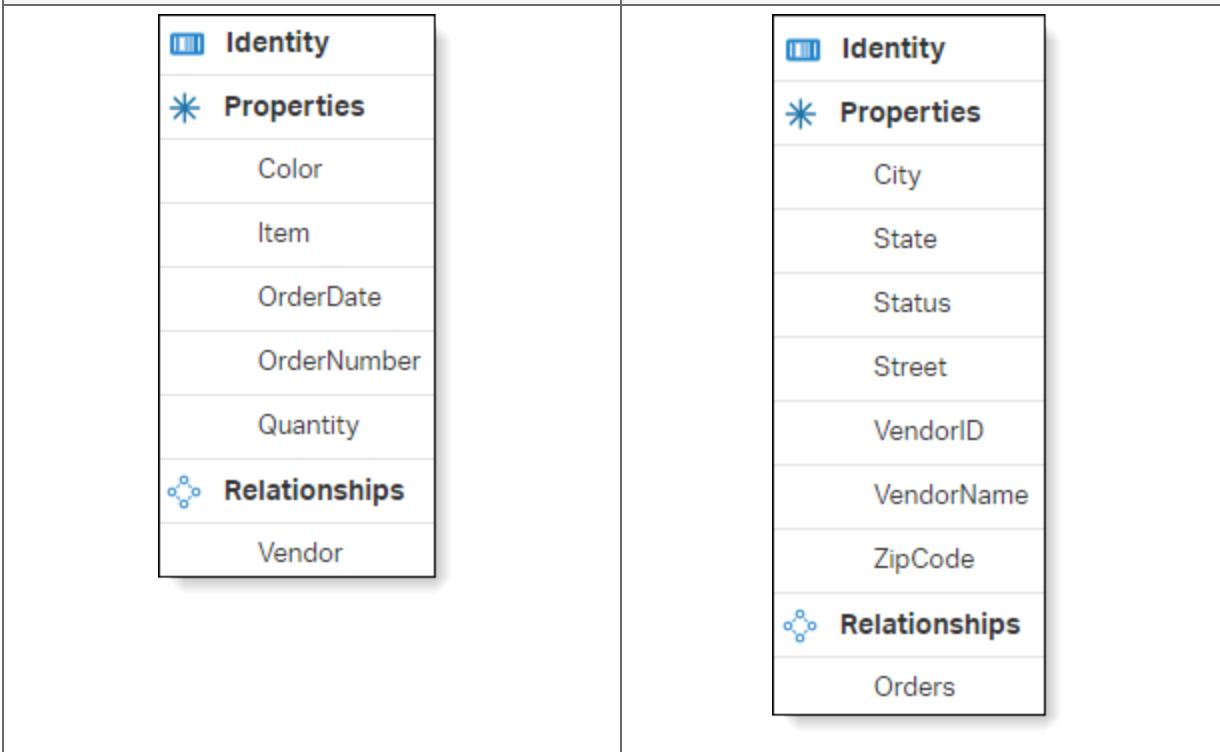
Advanced configuration

Length
64

5. In Advanced configuration, in Length, leave the default value. (Do not change any of the other values).
6. Click  (Save).

Optional. To verify your work, open Workspace Documents, right-click the **Order Management** project, and then select **Validate**.

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

Order building blocks	Vendor building blocks																																												
 <table border="1"> <thead> <tr> <th data-bbox="372 297 638 346">Identity</th> <th data-bbox="992 297 1241 346">Identity</th> </tr> </thead> <tbody> <tr> <td data-bbox="372 361 638 409">Properties</td> <td data-bbox="992 361 1241 409">Properties</td> </tr> <tr> <td data-bbox="372 424 638 473">Color</td> <td data-bbox="992 424 1241 473">City</td> </tr> <tr> <td data-bbox="372 487 638 536">Item</td> <td data-bbox="992 487 1241 536">State</td> </tr> <tr> <td data-bbox="372 551 638 599">OrderDate</td> <td data-bbox="992 551 1241 599">Status</td> </tr> <tr> <td data-bbox="372 614 638 663">OrderNumber</td> <td data-bbox="992 614 1241 663">Street</td> </tr> <tr> <td data-bbox="372 677 638 726">Quantity</td> <td data-bbox="992 677 1241 726">VendorID</td> </tr> <tr> <td data-bbox="372 741 638 789">Relationships</td> <td data-bbox="992 741 1241 789">VendorName</td> </tr> <tr> <td data-bbox="372 804 638 853">Vendor</td> <td data-bbox="992 804 1241 853">ZipCode</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th data-bbox="372 741 638 789">Relationships</th> <th data-bbox="992 868 1241 916">Relationships</th> </tr> </thead> <tbody> <tr> <td data-bbox="372 804 638 853">Vendor</td> <td data-bbox="992 910 1241 958">Orders</td> </tr> </tbody> </table>	Identity	Identity	Properties	Properties	Color	City	Item	State	OrderDate	Status	OrderNumber	Street	Quantity	VendorID	Relationships	VendorName	Vendor	ZipCode	Relationships	Relationships	Vendor	Orders	 <table border="1"> <thead> <tr> <th data-bbox="372 297 638 346">Identity</th> <th data-bbox="992 297 1241 346">Identity</th> </tr> </thead> <tbody> <tr> <td data-bbox="372 361 638 409">Properties</td> <td data-bbox="992 361 1241 409">Properties</td> </tr> <tr> <td data-bbox="372 424 638 473">Color</td> <td data-bbox="992 424 1241 473">City</td> </tr> <tr> <td data-bbox="372 487 638 536">Item</td> <td data-bbox="992 487 1241 536">State</td> </tr> <tr> <td data-bbox="372 551 638 599">OrderDate</td> <td data-bbox="992 551 1241 599">Status</td> </tr> <tr> <td data-bbox="372 614 638 663">OrderNumber</td> <td data-bbox="992 614 1241 663">Street</td> </tr> <tr> <td data-bbox="372 677 638 726">Quantity</td> <td data-bbox="992 677 1241 726">VendorID</td> </tr> <tr> <td data-bbox="372 741 638 789">Relationships</td> <td data-bbox="992 741 1241 789">VendorName</td> </tr> <tr> <td data-bbox="372 804 638 853">Vendor</td> <td data-bbox="992 804 1241 853">ZipCode</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th data-bbox="372 741 638 789">Relationships</th> <th data-bbox="992 868 1241 916">Relationships</th> </tr> </thead> <tbody> <tr> <td data-bbox="372 804 638 853">Vendor</td> <td data-bbox="992 910 1241 958">Orders</td> </tr> </tbody> </table>	Identity	Identity	Properties	Properties	Color	City	Item	State	OrderDate	Status	OrderNumber	Street	Quantity	VendorID	Relationships	VendorName	Vendor	ZipCode	Relationships	Relationships	Vendor	Orders
Identity	Identity																																												
Properties	Properties																																												
Color	City																																												
Item	State																																												
OrderDate	Status																																												
OrderNumber	Street																																												
Quantity	VendorID																																												
Relationships	VendorName																																												
Vendor	ZipCode																																												
Relationships	Relationships																																												
Vendor	Orders																																												
Identity	Identity																																												
Properties	Properties																																												
Color	City																																												
Item	State																																												
OrderDate	Status																																												
OrderNumber	Street																																												
Quantity	VendorID																																												
Relationships	VendorName																																												
Vendor	ZipCode																																												
Relationships	Relationships																																												
Vendor	Orders																																												

What's Next?

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

Create lists

An application uses one or more lists to filter and display a set of items based on your configuration. A 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 is presented. Lists are the most common way for a user to interact with an application. The user opens a list of items, and 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 later when you design forms to create and display orders and vendors.

In the application, the Label that you enter is shown as the name of the list. If multiple users are working on the same server, include your name or other distinguishing information in the Label to avoid having duplicate lists with the same name. For example, use All Orders Ben and All Vendors Ben as the Labels.

To create a list of orders:

1. In Workspace Documents, open the Order entity.
2. Navigate to Available building blocks > Presentation, and then click **List**.
The Add list dialog box opens.
3. In the Add list dialog box, in Label, type **All Orders**. In Name, the Label value is filled by default.
4. Click **Add**.
The list designer opens.
5. On the Properties tab, in Available Properties > Properties, select the properties that application users must see in the list:
 - Color
 - Item
 - Order Date
 - Order Number
 - Quantity
6. Expand **Vendor**.
7. In Vendor > Properties, select **Vendor Name**.
8. In the Properties shown in results pane, click **Collapse All**, click each property, 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 the application.

The screenshot shows the List Designer interface. On the left, the 'Available Properties' pane lists properties grouped under 'Identity' (Entity type, ID, Item ID, Item status), 'Properties' (Color, Item, Order Date, Order Number, Quantity), and 'Vendor' (Identity, Properties). The 'Properties' group under 'Vendor' includes City, State, Status, Street, Vendor ID, Vendor Name, and Zip Code. On the right, the 'Properties shown in results' pane displays the selected properties: Order Number - Text, Order Date - Date, Item - Text, Quantity - Integer, Color - Enumerated Text, and Vendor Name - Text. Below these are buttons for 'Expand All', 'Collapse All', 'Remove', 'Move Up', and 'Move Down'.

9. Click (Save) and close the window.

To create a list of vendors:

1. In Workspace Documents, open the Vendor entity.
2. Navigate to Available building blocks > Presentation, and then click **List**. The Add list dialog box opens.
3. In the Add list dialog box, in Label, type **All Vendors**. In Name, the Label value is filled by default.
4. Click **Add**.
The list designer opens.
5. On the Properties tab, in Available Properties > Properties, select the properties that application users must see in the list. Select them in the following order to avoid rearranging them later:
 - Vendor Name
 - Vendor ID
 - Status
 - Zip Code
6. In the Properties shown in results pane, click **Collapse All**. The properties are added to the Properties shown in results pane in the order in which you selected them.

The screenshot shows the List Designer interface. On the left, the 'Available Properties' pane lists several categories: Identity (Entity type, ID, Item ID, Item status), Properties (City, State, Status, Street, Vendor ID, Vendor Name, Zip Code), and Orders. On the right, the 'Properties shown in results' pane displays the selected properties in the order they were chosen: Vendor Name - Text, Vendor ID - Text, Status - Enumerated Text, and Zip Code - Text. Below these are buttons for Expand All, Collapse All, Remove, Move Up, and Move Down.

7. Click (Save) and close the window.

Optional. To check your work, open Workspace Documents, right-click the **Order Management** project, and then select **Validate**. Correct any errors that are found and then close the window after validation is successful.

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 two forms for the Order entity and two for the Vendor entity:

- Create - Users access this form to add an order or vendor to the system.
- Default - Users access this form to view, change, or delete a previously created order or vendor.

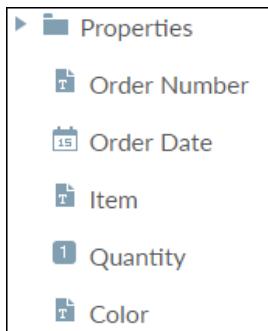
Since you created relationships between the Order and Vendor entities earlier, you can select properties from both entities when creating forms. In the Create form of the Order entity, you will include a browse icon (🔍) for users to select the vendor for the order. You will also include the related Vendor 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. Navigate to Available building blocks > Presentation, and then click **Form**.
The Add form dialog box opens.
3. In the Add form dialog box, in Label, type **Create**.

Important: This form, and any other form that is used to create items, must be named Create and must begin with an uppercase letter.

4. Click **Add**.
The form designer opens.
5. In the Components pane, drag the **Properties** heading for orders to the Presentation pane. You can alternatively drag the properties to the form one at a time.
6. In the Components pane, click the **Color** property, and then in the Color details pane, in Presentation > Type, select **Drop list**.

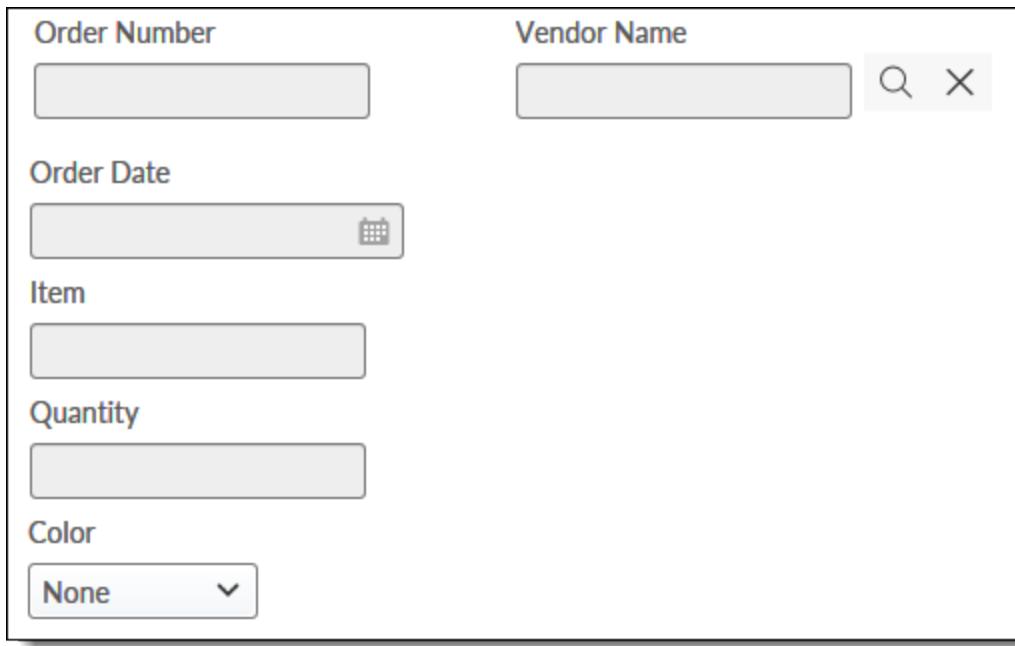


7. In the Components pane, from the Relationships list, drag the **[0..1] Vendor** relationship to the Presentation pane.



A browse icon (🔍) is placed in the Presentation pane.

8. In the Presentation pane, click  (Browse), and in the Vendor details pane, configure the Browse icon.
 - In Presentation > Browse list, select **All Vendors**. This specifies the list that is displayed when a user clicks  (Browse) to select a vendor.
 - In Actions, select **Clear**. Leave the default value of Browse selected. Application users will use these icons to find and clear related vendors when creating orders.
9. In the Components pane, in the Relationships list, expand **[0 . . 1] Vendor**, and then from the Properties list, drag the Vendor Name property to the Presentation pane.
10. Resize and reposition the components in the Presentation pane as follows.



The screenshot shows a Microsoft Power Apps form builder interface. It contains five input fields: 'Order Number' (text box), 'Vendor Name' (text box with a search icon and an 'X' button), 'Order Date' (date picker), 'Item' (text box), and 'Quantity' (text box). Below these is a 'Color' field, which is a dropdown menu currently set to 'None'.

11. Click  (Save) and then close the window.

Optional. To check your work, open Workspace Documents, right-click the **Order Management** project, and then select **Validate**. Correct any errors that are found and then close the window after validation is successful.

To build the form for creating vendors:

1. In Workspace Documents, open the Vendor entity.
2. Navigate to Available building blocks > Presentation, and then click **Form**. The Add form dialog box opens.
3. In the Add form dialog box, in Label, type **Create**. The form must be named Create and must begin with an uppercase letter.

4. Click **Add**.
The form designer opens.
5. In the Components pane, drag the **Properties** heading for vendors to the Presentation pane.
6. Resize and reposition the components in the Presentation pane 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 type="text"/>
<input checked="" type="radio"/> Active	Zip Code
<input type="radio"/> Inactive	<input type="text"/>

7. Click  (Save) and then close the window.

Optional. To check your work, open Workspace Documents, right-click the **Order Management** project, and then select **Validate**. Correct any errors that are found and then close the window after validation is successful.

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

1. In Workspace Documents, open the Order entity.
2. Navigate to Available building blocks > Presentation, and then click **Form**.
The Add form dialog box opens.
3. In the Add form dialog box, in Label, type **Default**.
4. Click **Add**.
The form designer opens.
5. In the Components pane, drag the **Properties** heading for orders to the Presentation pane.
6. In the Components pane, click the **Color** property, and then in the Color details pane, in Presentation > Type, select **Drop list**.
7. In the Components pane, from the Relationships list, drag the **[0..1] Vendor** relationship to the Presentation pane.
A Browse () icon is placed in the Presentation pane.

8. In the Presentation pane, click  (Browse), and in the Vendor details pane, configure the Browse icon.
 - In Presentation > Browse list, select **All Vendors**.
This list will be displayed when a user clicks  (Browse) to find a vendor.
 - In Actions, select **Clear**. Leave the default value of Browse selected.
Application users can use these icons to change the vendor for an order.
9. In the Components pane, in the Relationships list, expand **[0 . . 1] Vendor**, and then from the Properties list, drag the Vendor Name property to the Presentation pane.
10. Resize and reposition the components in the Presentation pane as follows:

Order Number	Quantity
<input type="text"/>	<input type="text"/>
Order Date	Color
<input type="text"/>	<input type="button" value="None"/> <input type="button" value="▼"/>
Item	Vendor Name
<input type="text"/>	<input type="text"/> <input type="button" value="Q"/> <input type="button" value="X"/>

11. Click  (Save) and then close the window.

Optional. To check your work, open Workspace Documents, right-click the **Order Management** project, and then select **Validate**. Correct any errors that are found and then close the window after validation is successful.

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

1. Open the Vendor entity.
2. Navigate to Available building blocks > Presentation, and then click **Form**.
The Add form dialog box opens.
3. In the Add form dialog box, in Label, type **Default**.
4. Click **Add**.
The form designer opens.
5. In the Components pane, drag the **Properties** heading for vendors to the Presentation pane.
6. In the Components pane, click the **Status** property, and in the Status details pane, in Presentation > Type, select **Drop list**.
7. Resize and reposition the components in the Presentation pane as follows:

Vendor Name	Street
<input type="text"/>	<input type="text"/>
Vendor ID	City
<input type="text"/>	<input type="text"/>
Status	State
Active	<input type="text"/>
	Zip Code
	<input type="text"/>

8. Click (Save) and then close the window.
9. To check your work, open Workspace Documents, right-click the **Order Management** project, and then select **Validate**. Correct any errors that are found and then close the window after validation is successful.
10. Publish your project by right-clicking **Order Management**, and then selecting **Publish to organization**. Correct any errors that are shown and then close the window when publishing is successful.

What's Next?

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

Test your application

At this point, you are ready to test your application. To get started, run the application. The application opens in the default runtime environment that comes with AppWorks Platform.

If it isn't already running, start the runtime environment in a new tab. When the application starts, your default home page opens. You can select Home Page from the list in the header area at any time to return to where you started. The default home page shows the lists of items that users created and the items in the selected list.

- If you have not created orders or vendors, the lists of orders and vendors will not contain any items.
- If you created orders and vendors using a previous version of the Order Management Quick Start tutorial, they are shown in the All Orders and All Vendors lists. In a shared environment, there could be many lists. In the following screenshot, the All Orders list was selected in the Lists panel and the All Orders Ben panel shows the orders that were created.

Order Number	Order Date	Item	Quantity	Color...	Vendor Name
A-456	2/16/2016	Amp Meter	5	Red	Acme Automation
B-456	2/16/2017	Barometer	3	Blue	Bell Industries
C-456	2/16/2016	Computer	1	Green	Capital Computers
D-456	2/16/2016	Desk	1	None	Danforth Desks

Clearing the cache and refreshing the display

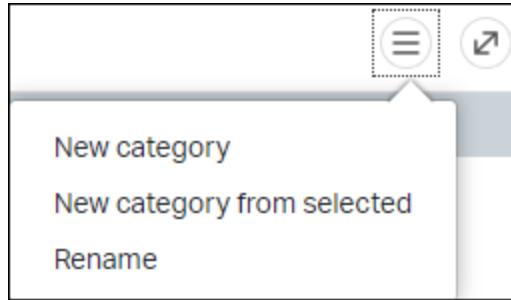
To clear the cache and refresh the display to see your recently-published changes, ensure that you press CTRL+F5 when the application starts.

Categorizing your lists

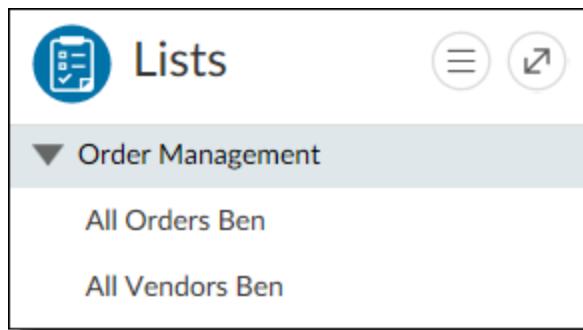
Although it is not a requirement, it can be very helpful to organize your lists into one or more categories to make lists from the same application available from a single location. This is especially useful if there are many lists or similarly-named ones in the Lists panel. Otherwise, the lists are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

To create a category for your lists:

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



2. In the text box that opens, type your name (or other text to identify the category).
3. Click (Save).
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. For convenience, you may want to drag the category to the top of the Lists panel.
If you click a list but you have not created any orders or vendors, 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 a vendor:

1. Click **+** (Create a new item) in the header area, navigate to All items, and then click **Vendor** (<your name>).

The screenshot shows a 'Create Vendor' form. It has two columns of input fields. The left column contains 'Vendor Name' (with a single character entered), 'Vendor ID' (empty), and 'Status' with three radio button options: 'Active' (selected), 'Inactive', and 'None'. The right column contains 'Street' (empty), 'City' (empty), 'State' (empty), and 'Zip Code' (empty). All fields are represented by simple text input boxes.

2. Enter information about the vendor, and then click **Save and create another**.

Important: Enter compatible values for the city, state, and zip code. For example, if you enter Atlanta, GA as the city and state, look up a zip code that is relevant for that location. All other values can be fictitious.

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 an order:

1. Click **+** (Create a new item) in the header area, navigate to All items, and then click **Order (<your name>)**.

The screenshot shows a 'Create Order' dialog box with the following fields:

- Order Number:** An input field containing a single character 'I'.
- Vendor Name:** A search bar with a magnifying glass icon and an 'X' button.
- Order Date:** A date picker input field.
- Item:** An input field.
- Quantity:** An input field.
- Color:** A dropdown menu showing 'None'.

2. Enter information about the order in Order Number, Order Date, Item, Quantity, and Color.
3. Click **?** (Browse) for Vendor Name.
4. Select a vendor from the list, and then click **Select**. The list of vendors will show the vendors you added.

>	Vendor Name	Vendor ID	Status	Zip Code
<input type="radio"/>	Acme Automation	A-123	Active	03031
<input type="radio"/>	Bell Industries	B-123	Active	06108
<input type="radio"/>	Capital Computers	C-123	Active	77036
<input type="radio"/>	Danforth Distributors	D-890	Active	02365

5. Click **Save and create another**.
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: To view all the columns in a single row, increase the window size. To perform toggle sorting for a column in ascending or descending order, click the column heading.

<input type="checkbox"/>	Vendor Name	Vendor ID	Status	Zip Code
<input type="checkbox"/>	Acme Automation	A-123	Active	03031
<input type="checkbox"/>	Bell Industries	B-123	Active	06108
<input type="checkbox"/>	Capital Computers	C-123	Active	77036
<input type="checkbox"/>	Danforth Distributors	D-890	Active	02365

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 perform toggle sorting for a column in ascending or descending order by clicking the column heading.

<input type="checkbox"/>	Order Number	Order Date...	Item	Quantity	Color...	Vendor Name
<input type="checkbox"/>	A-456	3/22/2018	Amp Meter	5	Red	Acme Automation
<input type="checkbox"/>	B-456	3/8/2018	Barometer	3	Blue	Bell Industries
<input type="checkbox"/>	C-456	3/16/2018	Computer	1	Green	Capital Computers
<input type="checkbox"/>	D-456	2/27/2018	Desk	1	None	Danforth Distributors

To view or edit a vendor or order:

- Select the check box for an item in the All Vendors or All Orders list, and then click **Open**, or click anywhere in the row. 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 All Vendors or All Orders list, and then 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 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. Navigate to Available building blocks > Business logic, and then click **Rule**.
The Add rule dialog box opens.
3. In Type, leave the default selection of Event to trigger the rule when an event occurs.
4. In Name, type **FutureDate**. The name must not contain spaces or invalid characters.
5. Click **Add**.
The rule designer opens.
6. 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, and then click **Done** to close the calendar. Notice that the current date is supplied as an absolute value.

Event

When A property changes ▾

ⓘ This rule will run on loading a form or when a property used in the following condition changes.

Condition

Basic Advanced

If all of the following are true

OrderDate greater than 04/22/2019

Action

Then Show warning ▾

Order date is greater than the current date.

The screenshot shows a rule configuration interface. The 'Event' section has 'When A property changes' selected. A note below it says 'This rule will run on loading a form or when a property used in the following condition changes.' The 'Condition' section has tabs for 'Basic' and 'Advanced'; 'Basic' is selected. It shows a condition 'If all of the following are true' with one item: 'OrderDate greater than 04/22/2019'. The 'Action' section has 'Then Show warning' selected, with a note below it saying 'Order date is greater than the current date.'

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

7. Click the **Advanced** tab and (as shown in the following screenshot) change the condition statement to:

```
item.Properties.OrderDate>today
```

The screenshot shows the 'Rule Designer' dialog box. It has three main sections: 'Event', 'Condition', and 'Action'.
Event: The 'When' dropdown is set to 'A property changes'. A tooltip explains: 'This rule will run on loading a form or when a property used in the following condition changes.'
Condition: The 'Basic' tab is selected. The condition expression is 'item.Properties.OrderDate > today'. A note below says: 'You have created an expression that cannot be represented in basic mode.'
Action: The 'Then' dropdown is set to 'Show warning'. The action message is 'Order date is greater than the current date.'

8. Click (Save) and close the window.

To configure the Disable rule:

1. Open the Order entity if it is not already open.
2. Navigate to Available building blocks > Business logic, and then click **Rule**. The Add rule dialog box opens.
3. In Type, leave the default selection of Event to trigger the rule when an event occurs.
4. In Name, type **DisableColor**. The name must not contain spaces or invalid characters.
5. Click **Add**. The rule designer opens.
6. Configure the rule as shown.

Event

When A property changes ▾

ⓘ This rule will run on loading a form or when a property used in the following condition changes.

Condition

Basic Advanced

If all ▾ of the following are true

Item equal to Laptop

Action

Then Disable

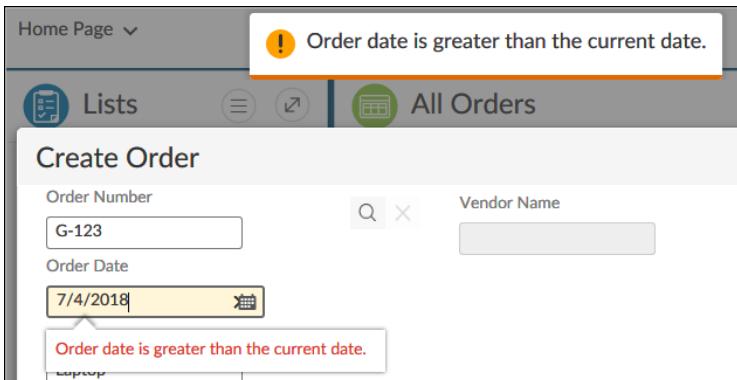
Property Color

The screenshot shows a rule configuration window with three main sections: Event, Condition, and Action. In the Event section, 'When' is set to 'A property changes'. A tooltip explains that the rule runs on loading a form or when a specific property changes. In the Condition section, 'If all' conditions are true, it checks if 'Item' is 'equal to' 'Laptop'. In the Action section, 'Then' the action is 'Disable', and the target is 'Color'.

7. Click (Save) and then close the window.
8. Right-click the **Order Management** project, and select **Publish to Organization**. Correct any errors that are shown and then close the window.

To test your rules in the application:

1. Open the application. 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 (which will differ from that shown in the sample screenshot) and verify that the warning you configured in the rule is displayed as a tooltip and at the top of the window.



3. Create an order and type **Laptop** (exactly as shown) in the Item field.
4. Click outside of the **Item** field and verify that the **Color** list is disabled.

What's Next?

In the next exercise, you will create customized layouts for viewing information about orders and vendors.

Create item layouts

Important: If you are using this exercise with an Order Management application that you created with a version earlier than AppWorks Platform 16.4, you must perform a few minor tasks before you begin. See [Upgrading the Quick Start application](#).

So far, you have used the Default form to display an item that you open from a list. In this exercise, you will create a customized layout to display information about a vendor that you select in your application. The layout will display the item information using the Create form that you designed for [creating a vendor](#). It will also use Google maps to show a web page of the area indicated by the vendor's zip code.

To make things more interesting, the layout will also provide a message board where users can collaborate about a selected vendor. To do this, you must add a Discussion building block to the Vendor entity.

You will also configure a layout for the Order entity. You will not use this layout in the current exercise, but you will need it when you create a home page in a later exercise.

To add a Discussion building block to the Vendor entity:

1. In the workspace, open the Vendor entity.
2. Navigate to Available building blocks > Functional, and then click **Discussion**.
The Discussion building block is added to the entity. No additional configuration is necessary.
3. Click  (Save).

To create a layout for displaying an item from the All Vendors list:

1. In the workspace, open the Vendor entity.
2. Navigate to Available building blocks > Presentation, and then click **Layout**.
The Add layout dialog box opens.
3. In Name, type **VendorLayout**, and then click **Add**.
The layout designer opens.
4. From the Panels pane, drag a **Form** panel to the Desktop pane, and then configure its properties.
 - In General > Name, type **Vendor Details**.
 - In Chrome, select **Full**. Chrome refers to the decoration around sections and panels.
 - In Panel Properties > Form, select **Create**.
5. Drag a Web Content panel to the right of the Form panel and configure its properties. A highlighted border indicates where the panel will be dropped. The Form panel that you already included in the layout is automatically resized to accommodate the new panel. You can also drag the blue divider to resize the panels.
 - In Name, type **Vendor Zip Code Area**.
 - In Chrome, select **Full**.
 - In URL, type the following text:

```
https://www.google.com/maps/embed/v1/place?key=AIzaSyBAp09cko7vcLvtk7ASRIZTrQD5MX
9xFNs&q={Item.Properties.City} {Item.Properties.ZipCode}
```

Important: Ensure that the URL is typed exactly as shown. Copying and pasting may introduce extra characters (such as amp for &) resulting in a syntax error in your application.

6. Drag a Discussions panel to the Desktop pane, position it below the Form panel, and then select **Full** in Chrome.
7. Click  (Save) and close the designer tab.
8. In the layout details pane, in Configuration, select the following options:
 - To view items in the preview panel
 - To view items in full screen

9. Click  (Save).

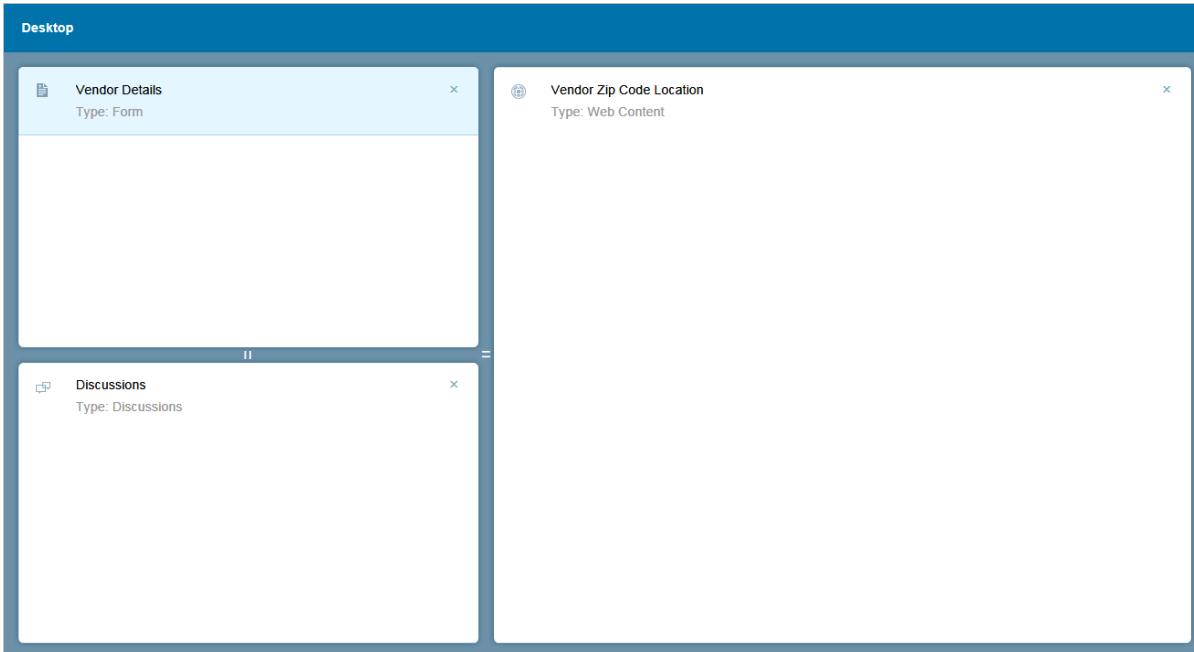
To add a Discussion building block to the Order entity:

1. In the workspace, open the Order entity.
2. Navigate to Available building blocks > Functional, and then click **Discussion**.
The Discussion building block is added to the entity. No additional configuration is necessary.
3. Click  (Save).

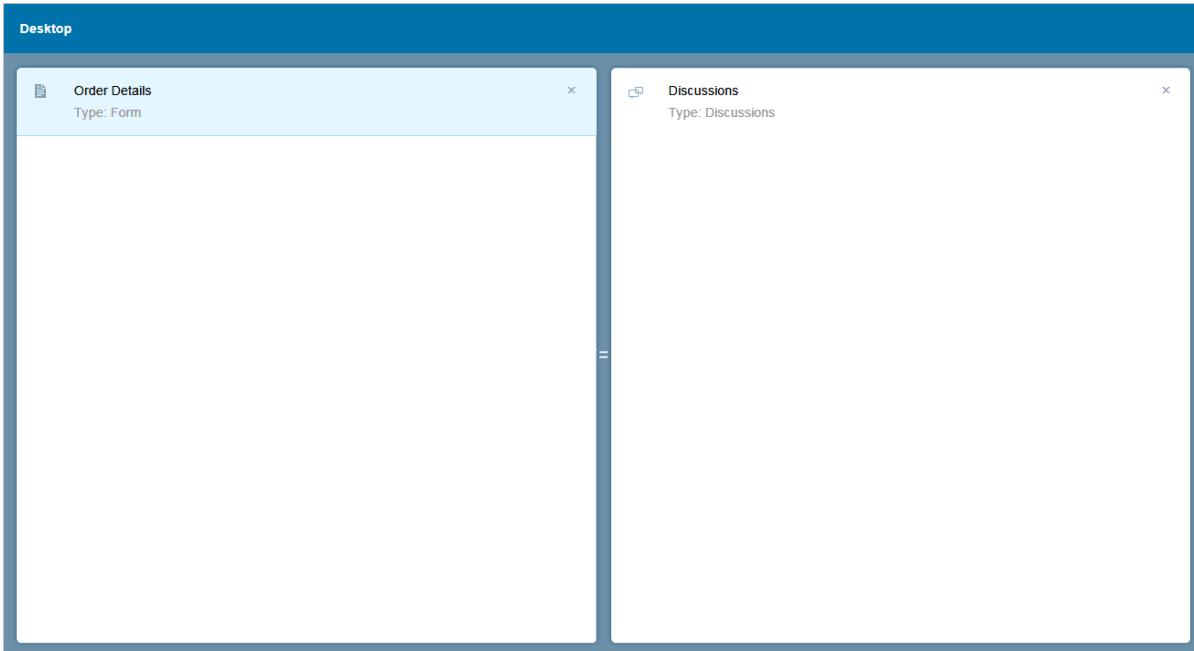
To create a layout for displaying an item from the All Orders list:

1. Open the Order entity if it is not already open.
2. Navigate to Available building blocks > Presentation, and then click **Layout**.
The Add layout dialog box opens.
3. In Name, type **OrderLayout**, and then click **Add**.
The layout designer opens.
4. From the Panels pane, drag a **Form** panel to the Desktop pane, and then configure its properties.
 - In General > Name, type **Order Details**.
 - In Chrome, select **Full**.
 - In Panel Properties > Form, select **Create**.
5. Drag a Discussions panel to the right of the Form panel, and then select **Full** in Chrome.
6. Click  (Save).
7. [Publish](#) the project and correct any errors that are found.

The Vendor Layout must look similar to the following example.



The Order Layout must look similar to the following example:



To test the Vendor Layout:

1. Open your application, refresh the display and, in the Home Page, expand the All Vendors list.
2. Select a vendor and click **Open** or double-click a vendor.
Details about the selected vendor, a google map, and an area where you can post a message must be displayed.

The screenshot shows two panels side-by-side. The left panel, titled 'Vendor Details', contains fields for Vendor Name (Danforth Distributors), Street (45 East 8th Avenue), Vendor ID (D-890), Status (Active), and Zip (02365). Below this is a 'Discussions' section with a post from 'cordys' about non-disclosure. The right panel, titled 'Vendor Zip Code Area', displays a map of New York City and surrounding areas, including Manhattan, Brooklyn, Queens, Bronx, and Staten Island, with various zip codes outlined in red.

What's Next?

In the next exercise, you will use the item layouts that you created in this exercise to create a customized home page.

Create a home page

So far, you have used the default home page (called Home Page) that comes with the application. In an actual business situation, you probably need different home pages for different types of users, roles, or departments.

Important: If you are using this exercise with an Order Management application that you created with a version earlier than 16.4, you must perform a few minor tasks before you begin. See [Upgrading the Quick Start application](#).

In this exercise, you will create a customized home page for users who are responsible for visiting vendors. The home page will include a Results panel that shows the items in the All Vendors list and a Preview panel that shows the [item layout](#) that you created for the Vendor entity.

You will also replace the logo in the home page with one that represents your organization. Before you begin, create a graphics file to use as the logo (since this is just for demonstration, it can be very simple). The recommended dimensions for the image are 55px in height and 385px in width. You will create a folder for images and then upload the image to that folder to make it available for the layout. To make the logo image available in your application, you will add a Web Library Definition to your project.

Note: If you cannot create graphics files, you can skip the procedures for creating a folder for images, including images in the folder, and adding a Web Library Definition to your project.

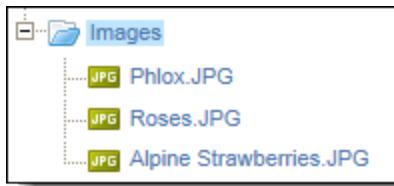
To create a folder for images:

1. Right-click the project and select **New > Folder** from the shortcut menu.
2. Replace Untitled Folder with **Images**, and then click **Save**.

To include images in the folder:

1. Right-click the **Images** folder, and then select **Upload Document**.
2. Click **Browse** and select the logo file.
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:

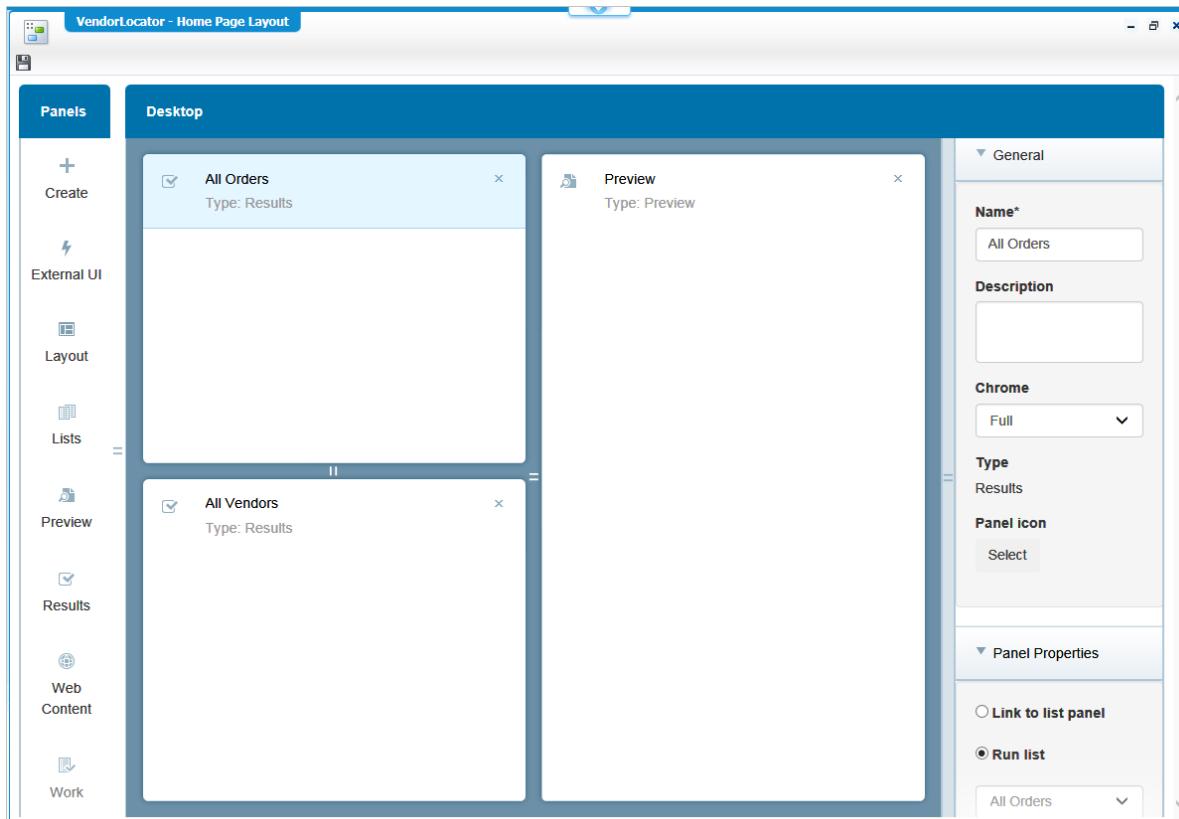
1. Open your workspace, if it is not already open, and select your project.
2. Click  (New).
3. Type **Web Library Definition** in the search box, and then click **Web Library Definition** in the search results.
You may also be able to select it by right-clicking your project and then selecting **New > Web Library Definition** from the shortcut menu.
4. In Name and Description, type **Web Library Definition**.
5. In Web Content Settings, select the **Images** folder, and then click **OK**.
6. Click  (Save) and close the window.

To create the home page:

1. In Workspace Documents, open your workspace.
2. Click  (New).
3. Type **home page** in the search box, and then click **Home Page Layout** in the search results.
4. In Display name, type **Vendor Locator**.
5. In Name, type **VendorLocator**.
6. In Layout type, ensure that Home page is selected.

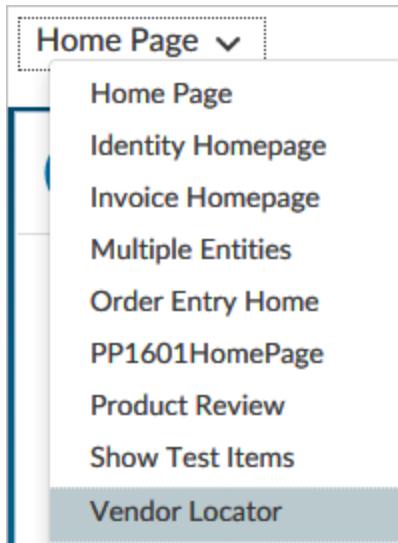
7. In Header image, click **Select**, and then browse to the image to use for the logo.
8. Click **Configure**. The panels that are available for adding to a home page are shown in the Panels pane.
9. Drag the Preview panel to the Desktop pane and accept the default settings (Chrome must be set to None).
10. Drag a Results panel to the Desktop pane and position it to the left of the Preview panel.
 - In Name, type **All Orders**.
 - In Chrome, leave the default value of None.
 - Select **Run list > All Orders**.
11. Drag another Results panel to the layout area and position it under the first Results panel.
 - In Name, type **All Vendors**.
 - In Chrome, leave the default value of None.
 - Select **Run list > All Vendors**.
12. Click  (Save).
13. Publish the project, correct any errors, and close the window when publishing is successful.

The Vendor Locator home page layout must look roughly like this.



To test the home page:

1. Open your application, refresh the display, and select the Vendor Locator home page.



2. Select three vendors. Each selected vendor is shown on a separate tab. When you select a vendor, the item layout that you defined for the Vendor entity is shown.

If you select an item in the All Orders list, the item layout that you defined for Orders is shown in a similar way.

Tip: The header area for this example contains a simple image. Your organization may want to employ a graphic artist to design the image.

The screenshot displays a user interface for the P&L Purchasing Department. On the left, there's a sidebar titled 'Vendor Locator' with sections for 'All Orders' and 'All Vendors'. Under 'All Orders', there are three entries: '2/2/2018 1234 Chair MacMillan Machinery', '6/14/2017 123 Table Phillips Products', and '6/14/2017 123'. Under 'All Vendors', there are four entries: 'Acme Enterprises A-123 Active', 'Danforth Distributors D-890 Active', 'EMC S876 Active', and 'Leonard Lawrence'. The main content area has tabs for 'Acme Enterprises', 'Leonard Lawrence', and 'MacMillan Machinery'. The 'Acme Enterprises' tab is active, showing 'Vendor Details' (Vendor Name: Acme Enterprises, Vendor ID: A-123, Status: Active, Street: 26 Spaulding Turn, City: Nashua, State: NH, Zip: 03031) and a 'Discussions' section with posts about the vendor's location and standards compliance. To the right of the vendor details is a map titled 'Vendor Zip Code Area' showing the location of Nashua, New Hampshire, with various roads and landmarks labeled.

What's next?

You are now ready to [explore](#) many additional capabilities of entity modeling and low-code design.

Copying a building block

Copying a building block provides the flexibility to reuse an existing building block. In this way, the original implementation of the building block becomes the starting point. You can create a building block, and copy it to create multiple ones with similar details and formatting. This reduces the time to create a building block with similar configurations.

Example

In the following example, there is a Form building block created for the manager. This form is then copied to create another one for the employee.

To copy a building block:

1. Create an entity or open an existing one.
2. Navigate to Available building blocks > Presentation and click **Form**.
3. Add a building block named Manager and add components to it in the designer.
4. Go back to the entity tab and point to the building block that you want to copy.

Note: If a building block can be copied, the Copy button is shown in the highlighted bar.

5. Click **Copy** in the highlighted bar to copy the Manager form.
The copied form is named CopyOfManager, and its details are displayed in the Details pane.
6. Rename this copy as **Employee** in the Details pane.
This copy contains all the components added to Manager.
7. Open the Form designer for Employee to adapt it to your needs.

Note: Changes you make to the original building block are not reflected in the copy.

Take the next step

Now that you have explored some basic entity modeling features, 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 specific 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](#)

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 [Adding building blocks to entities](#).

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

- To create and configure a single building block in one operation, define its attributes and then click **Add**.
- To create multiple building blocks and configure them later, click **Add and continue** after you define each building block and then immediately create another building block. Each building block that you create is added to the list of building blocks. You can configure each building block later by selecting it from the list.

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

- An application's entities, and their properties and relationships to each other, are called its domain model.
- Entities map to database tables.
- A project can contain any number of entities.

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

You use native entities to create an application whose data resides in the AppWorks 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 AppWorks. This is done through EIS connections that are available out-of-the-box with AppWorks 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 AppWorks Platform, you cannot change their structure.

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

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

- To create and configure a single building block in one operation, define its attributes and then click **Add**.
- To create multiple building blocks and configure them later, click **Add and continue** after you define each building block and then immediately create another building block. Each building block that you create is added to the list of building blocks. You can configure each building block later by selecting it from the list.

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 the application. For example, if the Display name is Order, the **Create new** list includes an option called **Order**.

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 for the name, display name, and description (in properties) is 128 characters.

Note: You can use an advanced concept called entity subtyping to help make large projects more manageable. See [Using subtyping for advanced modeling](#) for information about subtyping and about using the **Abstract** option.

5. If you want the entity to be available for use in other projects, select **Enable reuse in other applications**.

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. 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.

After you create at least one entity for a project you need to define security so that application users will be able to access your application. See [Defining security for 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.

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 specific 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.

Adding building blocks to entities

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. The order in which you can add building blocks to entities is not predefined. 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.

Building blocks are organized as follows in this documentation:

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

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.

- To create and configure a single building block in one operation, define its attributes and then click **Add**.
- To create multiple building blocks and configure them later, click **Add and continue** after you define each building block and then immediately create another building block. Each building block that you create is added to the list of building blocks. You can configure each building block later by selecting it from the list.

Important: The maximum length for the name, display name, and comments for a building block is 128 characters.

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 .
Identity	Enables synchronization of user information (identities) from Open Text Directory Services (OTDS). See Identity .
Relationship	Determines how entities are connected. Typically, the entities in a business domain are related to each other. See Adding relationships .

Building block	Description
History	Tracks specific changes or access to items to maintain a collection of history events. See Tracking history .
Tracking	Adds information to an entity to track when and by whom the item was created and last modified. See Tracking changes .
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 .
Sharing	Enables the sharing of individual items with someone else. See Configuring security for more information.

Process flow

Process flow building blocks define information about business processes. Also see [Integrating BPM processes and entities](#)

Building block	Description
Lifecycle	Defines how an item (entity instance) evolves as the data flows through the business. See Adding a lifecycle .
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 Adding an Activity flow .

Business logic

Business logic building blocks define business logic and integration with other parts of AppWorks 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 AppWorks Platform using web services. See Using web services .
Deadline	Specifies the schedule for completing an item. See Setting deadlines .
Assignee	Specifies entity responsibility, including the default assignee, whether email notifications are sent to assignees, and the email template to use for notifications. See Adding an assignee .

Presentation

Presentation building blocks define the user interface for the application. Also see [Adding home pages](#) and [Creating a theme](#).

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 item 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 .
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

Content building blocks provide features that enable application users to manage content.

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 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 .

Collaboration

Collaboration building blocks provide features that enable application users to communicate about items.

Building block	Description
Discussion	Facilitates collaboration among application users. See Adding a message board .

Building block	Description
Email	Enables use of email in an entity, including both sending and receiving emails from an entity instance. The Email building block has a child entity that has a Content building block in which the email attachments are stored. See Providing email functionality and Configuring the E-mail connector .
Email template	Creates templates that are available for application users to send emails. Creating an email template .
Inbox lists	Displays items from a user's AppWorks Platform Inbox in a list. This feature is provided by a shared application called Inbox Task Management, which is automatically installed when you install AppWorks. See Using Inbox lists .

Naming building blocks

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 building block name, display name, and description 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 Oracle 12.1 and earlier versions, for cross-database portability, property names can contain up to 30 bytes only. In English, one byte is equivalent to one character. Other languages might have different equivalencies. Consult your database administrator for details.

Do not use the following reserved words 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

Using toolbars and icons

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.



While creating an application, you can 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

Icon	Tooltip
	Align Bottom
	Align Left
	Align Right
	Align Top
	Browse
	Browse to icon
	Clear icon
	Clean Build Output (Workspace) Remove (Roles)
	Configure
	Configure Authentication
	Delete
	Delete selected components
	Fit to longest width
	Fit to shortest width
	Import Application Package
	Insert section after
	Insert section before
	Link (imported entity)
	Lookup
	Manage Database Configurations
	Move section down
	Move section up
	New
	Publish

Icon	Tooltip
	Quick Access Menu
	Refresh
	Revert
	Save
	Save All
	Show/Hide Entity Properties
	Switch Solution
	Switch Workspace
	Synchronize All
	Test Connectivity
#	Toggle grid
	Workspace Documents
	Workspace Properties Properties

Importing entities from database tables

By importing entities from existing database schemas, you can work with data from multiple applications and databases directly within the application. 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 application 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 an AppWorks Platform environment, you import each of your existing data structures from a relational database into the AppWorks 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 the application.

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 AppWorks Platform Database Metadata Modeler.

To set up AppWorks 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 the AppWorks Platform Advanced Development documentation.

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, click **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.

Note:

- If your database table schema changes after the initial import and you want to reflect the changes in the previously imported entities, you 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 the application.

Mapping data types

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

Data type	SQLServer	Oracle	PostgreSQL	Entity property data type
BIGINT	X		X	Big Integer
BINARY_DOUBLE		X		Double
BINARY_FLOAT		X		Float
BIT	X			Boolean
BOOLEAN			X	Boolean
CHAR/NCHAR	X	X	X	Text
CLOB/NCLOB		X		Long Text
DATE		X		Date and Time
DATE	X		X	Date
DATETIME	X			Date and Time
DECIMAL	X	X	X	Decimal
DOUBLE		X	X	Double
FLOAT(N)		X		Double
FLOAT(N)	X			Float for n<24 Double for 24<n<=53
INT/INTEGER	X		X	Integer
INT/INTEGER		X		Decimal
LONG		X		Long Text
NUMERIC	X	X	X	Decimal
NUMBER		X		Decimal

Data type	SQLServer	Oracle	PostgreSQL	Entity property data type
REAL	X		X	Float
REAL		X		Double
SMALLDATETIME	X	X	X	Date and Time
SMALLINT	X		X	Integer
SMALLINT		X		Decimal
TEXT/NTEXT	X		X	Long Text
TIMESTAMP		X	X	Date and Time
TINYINT	X			Integer
UNIQUEIDENTIFIER	X			Text
VARCHAR/NVARCHAR	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 the application, 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 can be created using the + (Create) action with the **in a dialog** option. Items can also be created with the **in a new row** option when the source table has an auto-increment primary key. See [Adding a To many relationship to a form](#).

- 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.
- You cannot relink an entity that was unlinked. Instead, you need 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 the information about translating text to a target language in the *AppWorks Platform Advanced Development* 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 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 selected employee record, and performing other functions.

Typically, you add properties to an entity first so that you can use them 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. Navigate to Available building blocks > Structure, and then click **Property**.
The Add property dialog box opens.
3. In **Label**, type a name to show in the application.
4. In **Name**, the Label value is filled by default.
5. In **Type**, select the type of data for the property.
6. Click **Add**.

Note: If you are adding more than one property, select **Add another** before clicking **Add**. When adding the last property, clear **Add another**.

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.

Label

The label 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 underscores (_). 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 introduces a duplicate property name will fail. See [Using subtyping for advanced modeling](#).
- For Oracle 12.1 and earlier versions, for cross-database portability, property names can contain up to 30 bytes only. In English, one byte is equivalent to one character. Other languages might have different equivalencies. Consult your database administrator for details.
- For a list of reserved words, see [Naming building blocks](#).

Type

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

The following types are available for adding to an entity.

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 a value. The first value (none) is automatically created but it is optional and you can delete it. The value and label must be unique across all items or an error message is displayed. The maximum length of the label 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

Type	Description
	it. The value and label must be unique across all items or an error message is displayed. The maximum length of the label 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 approximately -10^{38} to 10^{38} . Due to their representation, floating point numbers can lose precision due to rounding and must 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). Note: This property cannot be sorted using an Oracle database.
Text	A text value with a specified length limit.

Note: 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, drop list, or list box. 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 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 must 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 type of Text, although you intended to assign it a type of Date. If you delete the Sales Date property and then recreate it with the Date type (without first publishing the project), the database is not updated with the new type. Instead, you need to publish the project immediately after you delete the property to ensure that the Text type is deleted from the database and then recreate the Sales Date property with the Date data type.

The following types are available only for import.

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 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 must 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 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.
- 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.

Enable reuse in other applications

Specifies whether the property can be used in other applications. 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 [Using subtyping for advanced modeling](#)

Type-specific attributes

You can specify additional attributes, depending on the type. Some attributes become available immediately after you select the type, while others become available after you add the property and open it from the list of building blocks in the Added building blocks pane.

The following sections describe how to configure additional options for specific 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 label and default value:

1. In the Label 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 Maximum number of digits left of the decimal mark and Maximum 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 Minimum value and Maximum value, type the minimum and maximum value that a user can enter.
- In Default value, type the default value to be supplied.

Duration

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

Float, Integer

- In Minimum value and Maximum value, type the minimum and maximum value that a user can enter.
- In Default value, type the default value to be supplied.

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 Length, specify the maximum length of the values that will be entered in the list (for Enumerated Text only).
- In Label, enter a unique name to display for the **none** value.
- If you want to add a new value, click **Add** and then specify the Label and Value.

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 to remove, and then click  (Delete).
If you delete an item and republish the entity after items have been created in the application, 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 Move up or Move 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  (Browse to icon) in the row.

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  (Clear icon) in the row itself.

If you select Dynamic:**■ Process to retrieve all values with their labels**

This business process is required.

- If the business process was not already created, click  (Add) to create a business process with a dynamic enumeration contract, save it, and make the required modifications.
- If the business process was already created, select it and click **OK**.

See [The business process to retrieve the values and labels for one item](#) for details on implementing the business process.

■ Process to retrieve labels for the given values

This business process is optional. However, if your application requires labels for the values, you must provide this business process.

- If the business process was not already created, click  (Add), create a business process with a dynamic enumeration contract, save it, and make the required modifications.
- If the business process was already created, select it and click **OK**.

See [The business process to retrieve the labels for a list of values for details on](#) implementing the business process.

To clear the second business process:

- Click  (Clear).

See [Creating a business process for a dynamic enumeration](#) for additional information about how to configure dynamic enumerations.

Important: Effective with Release 16.3, business processes are closed by default. Therefore, each business process called from a dynamic enumerated property needs to have the required permissions. To configure security, right-click the business process and select **Define Runtime Security**.

Text

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 because they are reserved for future special purposes.

Type a default value with the pattern applied.

In the application, 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.

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) 999-99999'

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 the application, 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

To specify a pattern for a Text property:

- In **Pattern**, specify the text pattern to apply to the text.
- In **Default value**, specify the default text.

To filter results based on a property with a pattern, the user of an application must provide the actual value - not the value with the pattern.

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 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**.
4. Click **Add**.
5. If necessary, configure **Additional attributes** or **Type-specific attributes** for the property.
6. Click  (Save).

Creating a business process for a dynamic enumeration

A dynamic enumeration enables the retrieval of enumeration values from an external source. The value list is retrieved in the application 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 label. This process is optional. Labels are different from values, so if your application requires labels, you must provide this business process.

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

The business process to retrieve the enumeration values and labels for one item

If a dynamic enumeration property is used in a form or list, this 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 requirements for the namespace and the input and output messages are described in this section. Although it is recommended that you use the  (Add) icon to automatically create the process, the information in this section can be used to manually create the business process. In the following messages, replace [SolutionName] with the name of the solution and [EntityName] with 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. On the **General** tab, change the namespace to `http://schemas/[SolutionName]/[EntityName]/enumeration`.

Input message:

The input message is input for Business Process Management based on which output is generated. This is required only if the enumeration values depend on entity properties. The structure of the input message as follows:

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

To create the input message, you need to define the input schema fragment:

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]>
```

[SolutionName] - Name of the solution (for example: MyCompanyOrderManagement)

[EntityName] - Name of the entity (for example: Order)

[PropertyName] - Name of the entity property (for example: ShippingCountry)

Example:

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

3. Click **Finish** 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.

After you drag the XML schema fragment, the input message looks like this:

```
<GetEnumerationInputMessage xmlns
="http://schemas/MyCompanyOrderManagement/Order/enumeration">
    <Order xmlns="http://schemas/MyCompanyOrderManagement/Order">
        <ShippingCountry>PARAMETER</ShippingCountry>
    </Order>
</GetEnumerationInputMessage>
```

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

Output message:

The output message is output for Business Process Management. The structure of the output message is as follows:

```
<GetEnumerationOutputMessage xmlns
="http://schemas/[SolutionName]/[EntityName]/enumeration">
    <EnumerationValue>
        <Value></Value>
        <DisplayName></DisplayName>           <!--Optional field-->
        <IsDefault></IsDefault>           <!--Optional field-->
    </EnumerationValue>
</GetEnumerationOutputMessage>
```

The output of Business Process Management 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.

Example:

```
<GetEnumerationOutputMessage xmlns="http://schemas/MyCompanyOrderManagement/Order/enumeration">
    <EnumerationValue xmlns="http://schemas/MyCompanyOrderManagement/Order/enumeration">
        <IsDefault>True</IsDefault>
        <Value>Vendor1</Value>
    </EnumerationValue><br>
    <EnumerationValue xmlns="http://schemas/MyCompanyOrderManagement/Order/enumeration">
        <IsDefault>False</IsDefault>
        <Value>Vendor2</Value>
    </EnumerationValue>
</GetEnumerationOutputMessage>
```

The business process to retrieve the labels for enumeration values

This process is invoked to retrieve the labels for enumeration values, for example when multiple items are displayed in a list. The database stores only the values, not the labels. As the list of values for an item can depend on the context, invoking the first business process for a potentially large set of items could have a negative effect on performance. Therefore, a second business process is used to retrieve the labels for a list of values.

Use this process only if the enumeration values are different from the labels.

The requirements for the namespace and the input and output messages are described in this section. Although it is recommended that you use the  (Add) icon to automatically create the process, the information in this section can be used to manually create the business process.

Namespace:

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

Input message:

The input for this Business Process Management will be the output of the first Business Process Management.

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

Example:

```
<GetDisplayNameForEnumerationValueInputMessage xmlns
="http://schemas/MyCompanyOrderManagement/Order/enumeration">
    <EnumerationValue xmlns
="http://schemas/MyCompanyOrderManagement/Order/enumeration">
        <IsDefault>True</IsDefault>
        <Value>Vendor1</Value>
    </EnumerationValue>
    <EnumerationValue xmlns
="http://schemas/MyCompanyOrderManagement/Order/enumeration">
        <IsDefault>False</IsDefault>
        <Value>Vendor2</Value>
    </EnumerationValue>
</GetDisplayNameForEnumerationValueInputMessage>
```

Output message:

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

The `GetDisplayNameForEnumerationValueOutputMessage` contains 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`.

Example:

```
<GetDisplayNameForEnumerationValueOutputMessage xmlns
="http://schemas/MyCompanyOrderManagement/Order/enumeration">
    <EnumerationValue xmlns
="http://schemas/MyCompanyOrderManagement/Order/enumeration">
        <Value>Vendor1</Value>
        <DisplayName>Vendor1DisplayName</DisplayName>
    </EnumerationValue>
    <EnumerationValue xmlns
="http://schemas/MyCompanyOrderManagement/Order/enumeration">
        <Value>Vendor2</Value>
        <DisplayName>Vendor2DisplayName</DisplayName>
    </EnumerationValue>
</GetDisplayNameForEnumerationValueOutputMessage>
```

Notes:

- When you change the input schema of a business process used for a dynamic enumerated property, the system is not made aware of this until you update the configuration of the property itself, such as the label. Publishing the entity will then make the property aware of the changed business process input schema.
- 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` does not have a value because 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 labels 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 label.
- If the label retriever business process is provided, the value list retriever business process must also contain the `DisplayName` in the output message.
- The value list retriever business process and label retriever business process must always return the same label for a given value. Otherwise, the user may see inconsistent labels 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.
- When there is no display name retriever business process, the system shows the display names in Form and values in List from the value list retriever business process.
- If there is no label retriever business process, the system cannot show labels 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 label. If the value list retriever business process still returns labels, it would be inconsistent and confusing to show these labels 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.
- It is possible to use a dynamic enumeration property of a child entity on the parent entity's form.
- It is possible to use a dynamic enumeration property of a peer related entity on a form, but only if the property does not depend on the context. That is, it uses another property value to filter the enumeration values.
- To optimize performance, the response of the value list retriever business process and the label retriever business process are cached for a user. This cache is valid for 5 minutes. If modifications are made to one of the business processes, the new values are not immediately visible. Workarounds are to sign in 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 the following options:
 - Configure Monitoring on Process Level
 - Default Settings for all Activities
- If an entity or solution name is changed, namespaces in the business processes are not automatically updated. To update namespaces, click **Synchronize process namespaces**. After the namespaces are updated, this option is no longer visible. However, if the entity or solution name is being used in the business process implementation (for example, conditions, message maps, or so forth) changes, you must update accordingly.
- To enable filtering on a dynamic enumeration property which depends on the context, you must design the business process to return a fixed set of values when the context is empty.
- To enable filtering on a context-based dynamic enumeration property, design the BPM to return a fixed set of values when the context is empty.

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 types of properties. 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 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 added 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, but it is possible to define your own ID called Business ID.

Business ID

The business ID consists of a string followed by a sequence number.

To add your own business ID:

1. In Workspace Documents, open the entity.
2. In the Added building blocks pane, click **Identity**.
3. In the Identity pane, in Configuration, select **Add business ID**.
Additional fields needed for the definition are displayed.
4. In **Prefix** type a string of up to 20 characters. The string cannot start with a space character.
5. In **Minimum number of digits** type the minimum number of digits (between 1 and 20) to be shown. A number using fewer digits is padded with leading zeroes. A number using more digits is added completely (it is not truncated).

An example of the complete business ID is shown in **Example** based on your inputs.

Notes:

- You cannot add a business ID for imported entities and EIS entities.
- The business ID of a child entity is separate from, and not tied to, the business ID of a parent entity.
- The business ID is assigned when ItemStatus = 1 (see [Entity status](#)).
- The business ID generation starts when **Add business ID** is selected. The sequence numbering starts with 1.
- The business ID generation stops when **Add business ID** is cleared. All previously generated business IDs are removed.
- Changing a prefix or the minimum number of digits affects only new items. The sequence numbering continues.
- The business ID of a supertype entity also applies to all its subtype entities, as they currently share the same Identity building block. See [Using subtyping for advanced modeling](#).

Entity status

The Identity building block provides a property called ItemStatus. This property is used to implement draft items in a Create form in the application. When a user creates a new item, the ItemStatus property is set to 0 (Draft Item). When a user completes creating the item, the draft item is committed and the ItemStatus changes 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 the application 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 (such as order lines, customers, products, deliveries, and so forth) in addition to the main Order entity. 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 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 parent 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). It can be made a bidirectional relationship by combining it with a To many peer relationship called Reports that represents each supervisor's direct reports.

Relationship directions

A relationship can be made unidirectional or bidirectional by combining it with another relationship.

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

Two unidirectional relationships between two entities (in reverse directions) are not the same as a bidirectional relationship. The key difference is that two unidirectional

relationships do not influence each other. That is, both relationships are populated and unpopulated independently of each other.

With a bidirectional 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 related 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 related entity. If another instance of the related 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 related entity. 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 (unidirectional)	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 made bidirectional 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 (bidirectional)	Each instance of A may be made bidirectional with one instance of B, and you can navigate back and forth from either side. Some instances of A may not reference an

From A to B	From B to A	Description
		instance of B. Some instances of B may not be referenced by any instances of A.
To one	To many (unidirectional)	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 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 (bidirectional)	<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 produces 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 succeeds:</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 (unidirectional)	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 (bidirectional)	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.

From A to B	From B to A	Description
		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 related 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 made bidirectional 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 some 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. Navigate to Available building blocks > Structure and then click **Relationship**. The Add relationship dialog box opens.
3. In Label, type the name to show in the application.
4. In Name, the Label value is entered by default.

Note: 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
Unidirectional Peer	<ol style="list-style-type: none"> 1. Select Relationship type = relates to one or relates to many. 2. Click Browse, select the related entity, and then click OK. 3. Clear the Make this relationship bidirectional check box.
Bidirectional Peer	<ol style="list-style-type: none"> 1. Select Relationship type = relates to one or relates to many. 2. Click Browse, select the related entity, and then click OK. 3. Select Make this relationship bidirectional. A list is displayed if there are relationships available to make this relationship bidirectional.
Unidirectional Child	<ol style="list-style-type: none"> 1. Select Relationship type = has a child. The Browse button and Make this relationship bidirectional check box are no longer displayed. 2. Provide a Child label. The Child name is filled by default with the label value. Note: All child entities of the same parent must have unique Child names.
Bidirectional Parent Child	<ol style="list-style-type: none"> 1. Select Relationship type = has a parent. This option is available only for child entities in unidirectional relationships where Relationship type = has a child. This type of relationship can be added only once. <ul style="list-style-type: none"> ■ The Related entity is automatically set to the name of the parent entity. ■ The relationship is automatically made bidirectional with the has a child relationship from the parent entity.
Unidirectional or Bidirectional Reflexive	<ol style="list-style-type: none"> 1. Select Relationship type = relates to one or relates to many. 2. Click Browse, select the same entity as the source entity, and then click OK. <p>Note: To make the relationship bidirectional, select Make this relationship bidirectional. A list is displayed if there are relationships available to make this relationship bidirectional.</p>

6. Click **Add**.
7. Click  (Save).

To make a relationship available for use in other applications:

1. Select the relationship from the list of Added building blocks.
2. In Advanced configuration, select **Enable reuse in other applications**.
3. Click  (Save).

To specify the delete behavior of the related items:

1. Select the relationship from the list of Added building blocks.
2. In Advanced configuration, select or clear the **Allow users to delete a ... if referenced by a**
3. Click  (Save).

Deleting relationships

If a user deletes a related item or clears a relationship in the application (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 unidirectional To one and To many 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 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 **Allow users to delete a ... if referenced by a ...** option is selected, a reference remains when the related entity of the unidirectional (To one /To many) peer relationship is deleted. Also, the **A relationship changes** 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.

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

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

Note: To enable users to view the history, go to **Security editor** and give permissions to view history. See [Configuring security](#) for more information.

To add history to an entity:

1. In Workspace Documents, open the entity.
2. Navigate to Available building blocks > Functional, and then click **History**.
3. In Configuration, for History log, 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 .
 - The Untitled history log dialog box opens.
 - b. In **Display name** and **Name**, 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 .
 - f. If necessary, change the location for the history log, and then click **OK**.
 - g. Close the window.
4. If you want every user access to an item to be included, select **Include 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 .

To rename a history log:

1. Open the History building block.
2. In Configuration > 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 Configuration > History log, click  (Browse).

3. Click the down arrow next to the name of the history log and select **Delete**.
4. 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.

Showing relationship actions in history

The history display shows relationship actions. It displays the Title or Business ID, if there is one. If there is no Title or Business ID, it displays a shortened form of the Item ID. The hover text combines all available forms: Title, Business ID, and unique Item ID. This makes the history as meaningful and as user-friendly as possible, while still making the more detailed and technical information available, should it be needed to further investigate past events.

While the Business ID provides a consistent meaningful reference such as 'Case-001', the Title building block provides a flexible user-friendly means of identifying an item in the user's natural business context. While it may simply be text entered by the user, you would typically construct the Title by bringing together one or more of the item's properties through a translatable Title Specification.

When configured to use a Title Specification, the title the user sees in history is the current title, displayed in the current locale. If the title is composed, in part, by a field such as Description, when history is displayed the Title reflects the current description, not the description as it was when the history event was logged. The fact that the description was changed can, of course, be seen as a property change event in the appropriate place in history.

Tracking changes

The Tracking building block adds information to an entity to track when and by whom an 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.

You can add tracking 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 specific 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. Navigate to Available building blocks > Business logic and click **Tracking**.
The Tracking building block is added to the entity.

Note: 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.

3. Click  (Save).

Configuring security

Security is related to access control, permission to access specific entity data, to view data, to update data, or upload a document. Some users can create new entities, update these entities, and delete them. In other applications, a specific user or set of specific users can access, update, or delete a specific entity instance. This is different from accessing all instances of a specific entity type. Sometimes, when one or more users have access to a specific entity item, they may have to give other users access to this item to enable sharing.

Some applications may have the requirement to allow users based on specific conditions, such as specific security or access control requirements. For example, a sales representative can only update sales quotes for customers in their region.

Note: When an entity item is shared with a user, the user does not get default permission to manage the associated content or business workspace in the content repository. The administrator must provide permissions in the content repository for the user, otherwise, document management operations might fail.

There are different security mechanisms available for building an application. Following are some examples to determine the security controls to be used:

Roles-based access control

The default security model in AppWorks Platform is the Role Based Access Control (RBAC). This means that a user receives permissions to perform some tasks, like using a specific business process or viewing a specific list of entity items. In the context of entity modeling, a role can give access to the items of a specific entity. For example, the role OrderManager allows the user to see and update all Orders.

The role OrderManager is a classic AppWorks Platform role. This means that it can be used not only for entity security, but also for non-entity artifacts, such as business processes, web-services, and rules.

Use the Security building block to configure security for entities, where users require access to all items of a specific entity. You can configure security for entities by using roles in the Security designer.

When to use the Security building block:

- When all users cannot access all entity items. This implies that access control must be defined.
- When users must get access to all items of an entity through a role.
- When an administrator must manage end-user role assignments.
- When access control for entity items is configured in design-time, by defining permissions for different roles.
- When it is not required to control access to specific entity items.

Instance-specific entity permissions

There are situations where information can only be accessed by specific users. This includes situations where only individual users must have access to an entity item, or only a specific group of users can update an item.

Sample scenario in a law firm where legal matters are handled: Only some users can create matters. When creating a matter, only the person creating it can view, update, or delete it. When other users must be involved, the matter creator can give someone else access to this matter by sharing it. Now, only these two users can access this matter. Later, the user shares it with a group of people. Another group of users has read-only permission to all matters using a normal role, but for a specific matter, one of these users' access to the entity item must be blocked.

This example shows that there are situations where standard role security is not sufficient. Sometimes, defining access control at design-time is inadequate for users. Instance-specific entity permissions allow the end user to change access control for individual users at run-time, by assigning sets of permissions to other users. This enables the user to share access to the entity item with other users.

The permissions given to other users to access the entity item can vary based on the application. Different applications have different needs for permissions. For some applications, it may be sufficient to allow a user to view, update, or delete an entity item. For other applications, a broader spectrum of permissions must be allowed, like uploading and downloading documents, allowing use of specific lists and actions.

Instance roles

To ensure that the end user does not have to assign individual permissions to users, the concept of instance roles has been added to entity modeling. An instance role is similar to a normal role. However, unlike a normal role, an instance role can be used to define a set of permissions that can be given by one user to another when sharing an entity item. A normal role is managed and assigned to users by the administrator.

Like a normal role, an instance role is also created during application development. The application builder defines the permissions that are configured for a specific instance role. An instance role cannot have the create permission assigned, since they deal with permissions of specific, existing entity items.

The application builder can use the Sharing building block to assign permissions to an instance role.

Interaction between the Sharing and Security building block

Configuring sharing for an entity can be done by adding instance roles in the sharing designer and assigning permissions to them. When an instance role is added to the Sharing building block, it exposes a set of permissions. These permissions can be used in the Security building block and be assigned to the available instance roles.

The same feature is also available in the Sharing designer. With this, the application builder can specify that if an item is shared by an end user with another user using an instance role, the other user can share this item with a third user.

The end user can share entity items with other users by using the Share dialog box. This dialog box can be accessed using the Share action button. This button is visible when the Sharing building block is enabled for the related entity, and when the user has permission to share the entity item and view with whom the item is shared.

Runtime share panel

The Share panel displays the list of users with whom the entity item is shared. To share the item with another user, select the user and assign an available instance role.

The process of sharing an item with permissions is simplified for the end user, as the instance role is a predefined set of permissions.

For example, the application builder configures the following instance roles for an entity type:

- can_view: Provides a read-only view of the entity data
- can_edit: Allows reading and updating
- can_comment: Allows a user to add comments using the added Discussion building block
- can_add_documents: Allows a user to add documents using the added Content building block

Depending on the user's permission, and the instance roles they can assign, these instance roles are displayed in the Share dialog box.

The possibility to share an item with another user implies the ability to revoke it.

Note: Users can deny access to a specific file or folder using the Content panel. However, they cannot share individual files and folders using this panel, even if the Sharing building block is added to the Contents child entity.

When a user has the ability to deny access, it is also displayed in the Share dialog box. The permission to deny access is termed as **No Access** in the Share dialog box.

Note: The list of shared users does not include all users who can access the entity item. Only users who have an instance role assigned to them for this entity item are displayed. Users who can access the entity item using a normal role are not displayed in the list unless the specific item is explicitly shared with them.

Permission assignment cascading over entity relations

Entities often have a relationship with other entities, for example, an order is related to order lines, or a person is associated with addresses. Taking the example of order and order lines, when a user shares an order item with another user, they may want to enable them to access the order lines. They can manually scan through each order line and share it with the other user, but this may not be convenient. To simplify this process of sharing an entity and related data, the concept of cascading permission assignments over relationship is added to the instance-specific entity permission. According to this process, when a user shares an order item, the related order line items also get shared. For example, if the user grants another user access to the order item and assigns the 'can_view' permission, then due to the permission assignment cascading over relations, the other user also gets 'can_view' permission for all related order lines. Both permissions are now available, in addition to those that will be created in the future.

This cascading of permission assignments works for relationships of type to-child. The same instance role needs to be configured for both the parent and child entity. In this example, both order and order line must have a Sharing building block in which the 'can_view' instance role is configured.

Hence, the cascading feature is the membership of the user with the instance role that is cascaded over the to-child relationship from order to order line.

When to use the Sharing building block

- When access to entity data must be handled for each entity item, instead of each entity type.

- When an individual user can share and give access to an individual entity item.
- When an end user must determine to whom they want to give access to specific entity data.
- When a user must have temporary access to individual entity items.
- When a user must have access to a specific entity type, that is, all entity items of this entity type, but must not have access to some specific entity items.

Conditional permissions

Users are sometimes only allowed to access data based on a specific condition. For example, the Sales representative is only allowed to update sales quotes for customers in their own region. Here this role has a condition for the update permission on the SalesQuote entity. Sample condition: `if SalesQuote.Region == USER.toWorkAddress.Region`.

Conditional permissions can be used both, in the Security and Sharing building block.

See [Granting conditional permissions](#) for details on using conditional permissions with entities.

Default closed security configuration

When a solution is deployed, it is closed to all users by default. A solution is initially 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 on 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, application users can perform only those operations on that entity that are granted to them in the Security designer.

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 designer.

Entity Permission	Description
Create	Users can create new instances (items) of the entity.
View	Users can see that a specific item exists. This permission does not allow users to see the properties and relationships of the item. To do that, specific read permissions on the properties and relationships are needed. 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 designer.

Building block	Permission	Description
Activity flow	Execute	Adds a case model to an entity, thus converting the entity into a case. This building block enables specifying permissions for Activity flow.
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 manage workspace and content in the applications See Business workspace task permissions .
Content		Users can 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	Add comment	Enables actions for adding a comment. See Discussion task permission .
Email	Send email	Users can send email.
Email template	Use	Users can use a specific email template to

Building block	Permission	Description
		compose an email.
File		See File task permissions .
History	View history	Users can view history.
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 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 the application.
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 related entity instance via the source entity instance. Without Read permission on the relationship, the relevant related entity instance cannot be accessed from the source entity instance. This permission does not allow users to read properties or relationships of the related 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 related 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.

Building block	Permission	Description
		<p>This Update permission does not allow users to update properties or relationships of the related entity instance. To do that, specific update permissions on these properties and relationships are needed.</p> <p>Note: For bidirectional 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 bidirectional relationship (to many to entity B) with update permission. This means 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.</p>
Rule	Update	<p>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.</p>
Sharing	View users	These users can see which other users have access to a specific shared item.
	Deny users	These users can deny other users access to a specific item. This permission is displayed as No Access in the Share dialog box.
	Assign to users	This permission is added for each instance role available in the Sharing building block. It enables a user to share an item with another user and grant them the permissions configured for the instance role.
	Auto assign to creator	This permission is added for each instance role available in the Sharing building block. When enabled for a classic role, the system automatically assigns this instance role to the user who created the item.
Title	Update	These users can deny other users access to a specific item. This permission is displayed as No Access in the Share dialog box.
Tracking	Read	This building block adds two relationships,

Building block	Permission	Description
		CreatedBy and LastModifiedBy. To access the properties in these relationships, select this permission for the relationships in the Security building block.
	Update	This permission is added for each instance role available in the Sharing building block. It enables a user to share an item with another user and grant them the permissions configured for the instance role.
Web service	Use	This permission is added for each instance role available in the Sharing building block. When enabled for a Classic role, the system automatically assigns this instance role to the user who created the item.

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 designer. 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.

Action/permission	Description
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 designer.

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 attachment	Users can delete an attachment added to the entity and 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 designer. For the Content building block, the permissions can be configured at the 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 AppWorks Platform is configured for Content Server, users will see check in, check out, and other actions that are applicable to Content Server.

Important: To view the audit history of actions, users need the DocumentStore Admin role in User Manager. To open an item configured with the Content building block, Read permission is required on ChildNodes.

Action/permission	Description
View folder	Users can see the folder and its contents. This permission is required to perform any of the other actions described in this table.
Upload	Users can upload a file. Update permission is required on the Title for this action to work. Update permission is required on ParentFolder.
Download	Users can download a file attached to the entity.
Check in	Users can check in a version of the file. To perform this action, users also need View properties permission.
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. To perform this action, users also need Update properties permission.
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 available only when the document store connector is configured with Brava.
Delete document	Users can delete an uploaded document.

Discussion task permission

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

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 designer. 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 AppWorks 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. To perform this action, users also need View properties permission.
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. To perform this action, users also need Update properties permission.
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.
Delete versions	Users can delete a version of the document.

Action/permission	Description
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.
Delete document	Users can delete an uploaded document.

Lifecycle task permissions

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

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.
Properties	Users can view the properties of the task.
Resume	Users can resume a paused task.
Revoke	Users can revoke a claimed task.

Action/permission	Description
Skip	Users can skip a task. You can skip tasks before they are 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.

Activity flow permissions

The following table describes the permissions that can be configured on an activity flow using the Security designer.

Action/permission	Description
Activity flow - Static action	Users can plan an activity flow for an entity item.
Initiate Activity flow	Users can plan an activity flow for an entity item.

Setting permissions

To set permissions, determine the type of access control your application needs. See [Roles-based access control](#) and [Instance-specific entity permissions](#). For role-based access control, you must add the Security building block to the entity. For instance-specific entity permission, you need to add the Sharing building block. In some cases, you can add both building blocks. These building blocks enable you to define security for entities, their properties, relationships, actions, and other rights available for the various building blocks.

To add role-based security for an entity:

1. Open the Workspace Documents folder.
It must contain at least one entity.
2. Open the entity that needs role-based security.
3. Navigate to Available building blocks > Business logic, and then click **Security**.
The Security building block is added to the entity and the Security designer opens.
4. In the Security designer, grant permissions to roles.
The permissions are grouped for each building block added to the entity.
5. Click  (Save).

To create a role:

1. In the Roles panel, click  (Add a Role to define Permissions).
The Select a role dialog box opens.
2. Select a role from the available options or click **New > Role**.
3. **Optional.** If you are creating a new role, type a name and description for the role.
4. **Optional.** For the new role, 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 add instance-specific permission for an entity:

1. Open the Workspace Documents folder.
It must contain at least one entity.
2. Open the entity to which the sharing capability must be added.
3. Navigate to Available building blocks > Business logic, and then click **Sharing**.
The Sharing building block is added to the entity and the Sharing designer opens.
4. In the Sharing designer, grant permissions to instance roles.
Permissions are grouped for each building block added to the entity.
5. Click  (Save).

To create an instance role:

1. In the Roles panel of the Sharing designer, click  (Add a Role to define Permissions).
The Select an instance role dialog box opens.
2. Click **New > Instance role**.
3. Type a Name and Display Name for the instance role. The location is filled by default.
4. Click  (Save).
5. Click **OK**.

Note: The Security building block and Sharing building block use the same permission editor. The following procedures of handling permissions for roles is applicable for both roles and instance roles.

To grant permissions to a role in the Security designer:

1. In the Roles panel, select a role or click  (Add a Role to define Permissions) to create a role.

2. In the Permissions panel, select each permission to grant to the selected role.

Note: Use the check boxes in the table header row to 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. 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) and close the Security designer.

To update permissions to a role in the Security designer:

1. In the Roles panel, select the role.
2. In the Permissions panel, select or clear the permissions to change.
3. Click  (Save) and close the Security designer.

To delete the permissions for a role in the Security designer:

1. In the Roles panel, select the role.
2. Click  (Delete all permissions for the selected roles).
3. Click  (Save) and close the Security designer.

To refresh the defined permissions in the Security designer:

- Click  (Refresh) to refresh the list of building blocks and modified permissions. A message notifies you that refreshing the entity will discard all changes and prompts you to specify whether you want to continue.

To remove security from an entity:

1. In Workspace Documents, open the entity.
2. Position the pointer over **Security**.
3. Click  (Delete). A message prompts you to specify whether you want to delete the Security building block.
4. Click  (Save).

Granting 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.

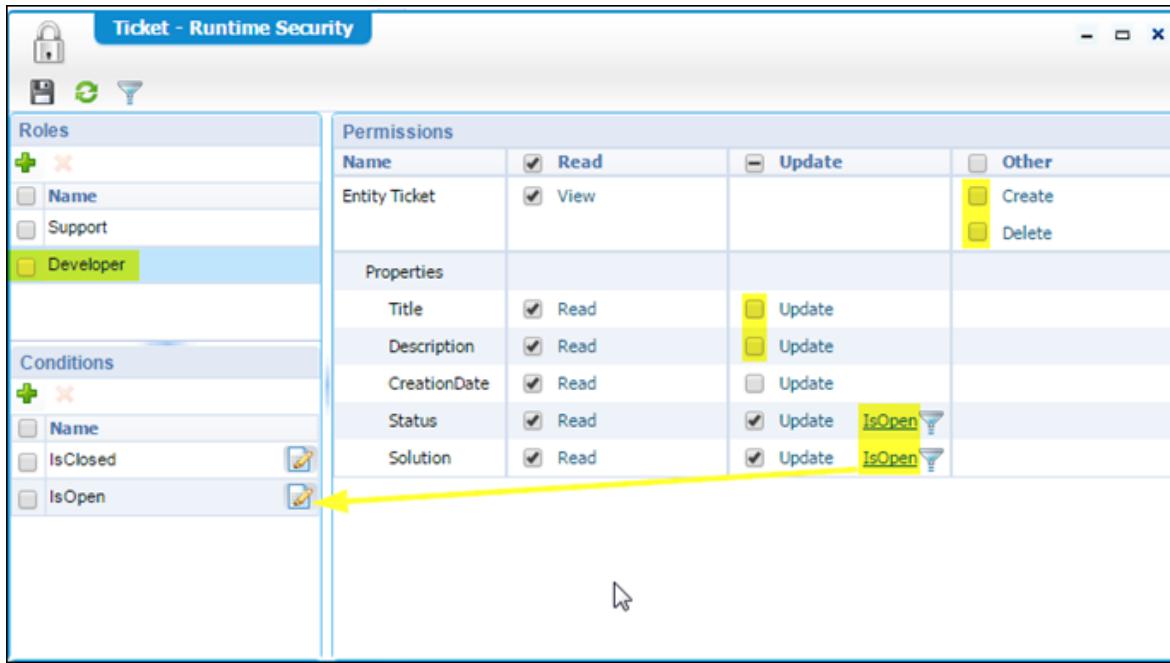
Example

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.

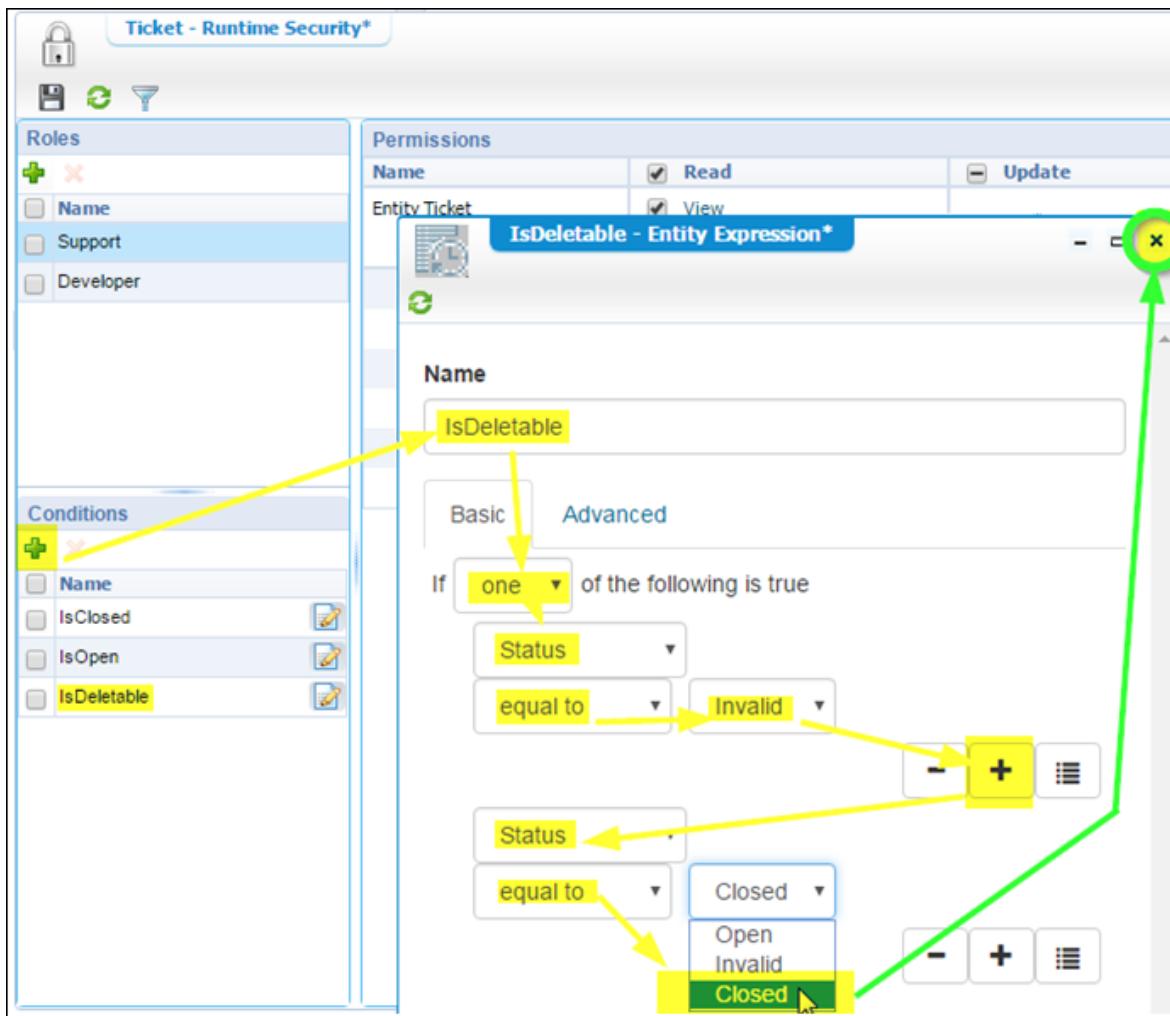
Name	Read	Update	Other
Entity Ticket	<input checked="" type="checkbox"/> View		<input checked="" type="checkbox"/> Create <input checked="" type="checkbox"/> Delete
Properties			
Title	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	
Description	<input checked="" type="checkbox"/>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

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.

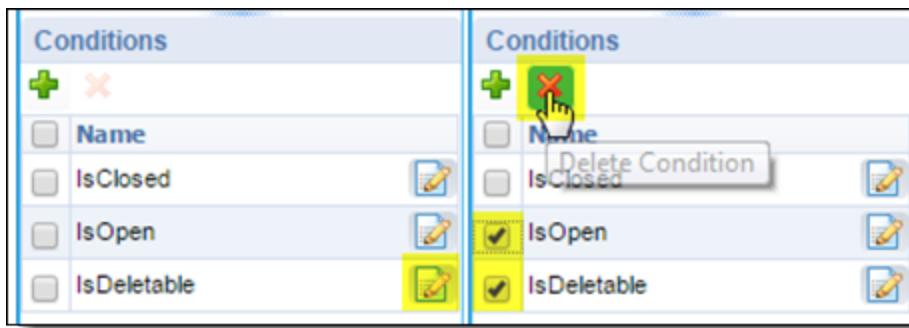


Notice that you need to click the (x) button in the upper right corner of the dialog box to finish the definition of the expression. The expression is saved as soon you click **Save** in the ACL Editor or in the Entity Modeler.

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).

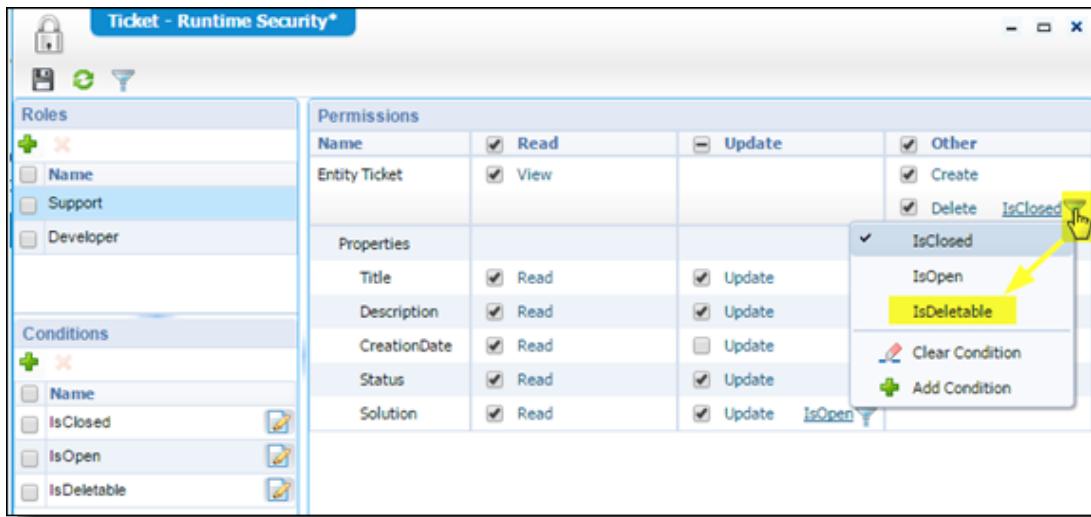


Assigning a condition to a permission

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.

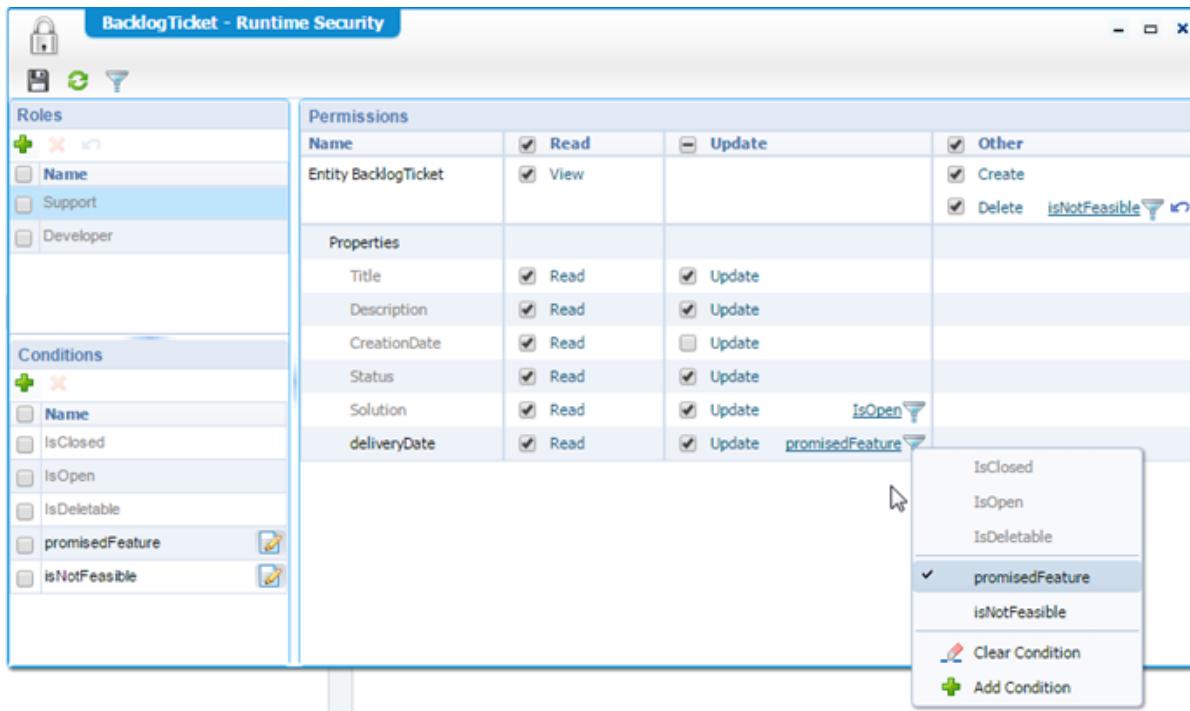
Name	Read	Update	Other
Entity Ticket	<input checked="" type="checkbox"/> View		<input checked="" type="checkbox"/> Create <input checked="" type="checkbox"/> Delete IsClosed
Properties			
Title	<input checked="" type="checkbox"/> Read IsOpen	<input checked="" type="checkbox"/> Update	
Description	<input checked="" type="checkbox"/> Read IsOpen	<input checked="" type="checkbox"/> Update IsOpen	
CreationDate	<input checked="" type="checkbox"/> Read IsClosed	<input type="checkbox"/> Update	
Status	<input checked="" type="checkbox"/> Read IsClosed	<input checked="" type="checkbox"/> Update IsClosed	
Solution	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update IsOpen	
Lists			
Untitled_List	<input checked="" type="checkbox"/> Use		

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.

Name	Read	Update	Other
Entity Ticket	<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 type="checkbox"/> Update	
Lists			
Untitled_List	<input checked="" type="checkbox"/> Use		

Customization and subtyping

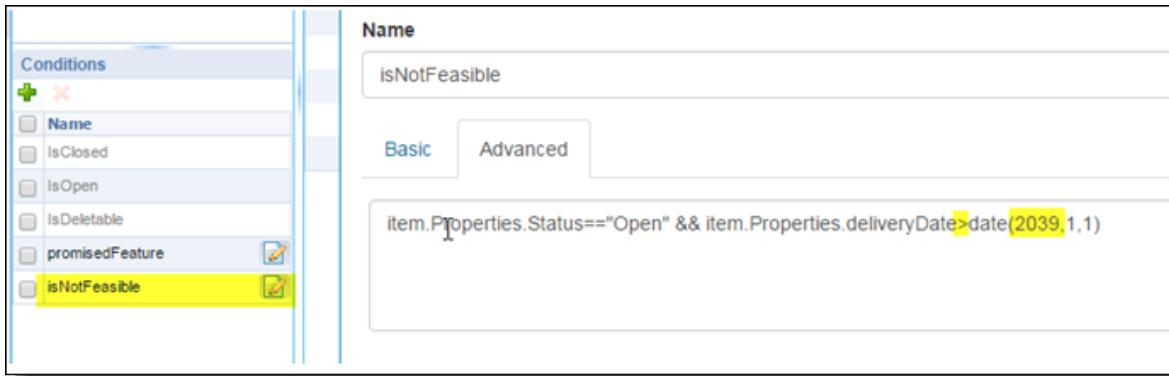
Assume that you create a subtype called BacklogTicket with an additional property called deliveryDate. That subtype entity inherits all the conditions defined on the supertype. The conditions from the supertype are indicated with light gray highlighting. The inherited conditions can be used, but cannot 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).



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

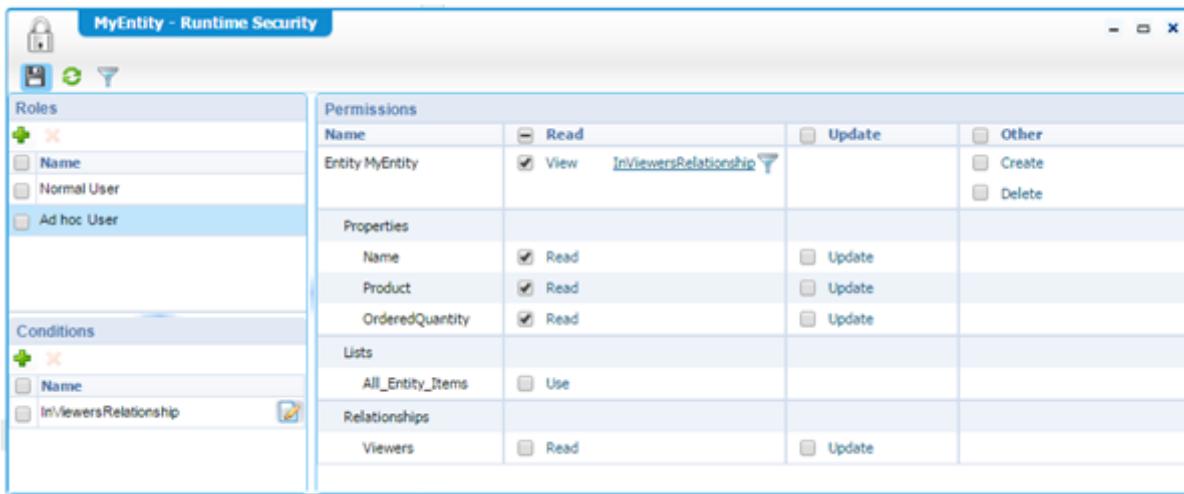
Use the Security building block to configure read-only permission for viewing entity data. To do this, you can configure permissions for a certain role in such a way that only read access to the applicable properties is granted. Any user who has that role assigned is able to view the entity data.

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 the application 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 related 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

There are three ways that you can integrate workflow processes into your low-code application.

- Lifecycle
- Activity Flow
- Business Process Model (BPM)

Lifecycle

Use a lifecycle for unstructured case management processes. A case management process is a process that doesn't presume that there is always an order and structure to work. This doesn't mean that there is no goal or structure to a lifecycle. Instead, it means that the knowledge worker may take different steps and complete tasks in a different order based on how and when events happen and their experience in resolving the business problem. A lifecycle enables knowledge workers to control the process and change it on a case-by-case basis. They have complete control over the case. For example, they can add tasks on the fly, assign tasks to participants, or re-route tasks to meet the needs of the case.

Lifecycles have the following characteristics:

- An entity can include only one lifecycle.
- Lifecycles provide more flexibility and advanced functionality than Activity Flows. For example, you can add stages or states to a lifecycle, leverage a business calendar, or trigger a BPM.
- Lifecycles can trigger a business process (for more structured automated work that doesn't need human interaction).
- Lifecycles can include ad hoc tasks that knowledge workers can add on the fly.
- Activities (tasks) can leverage entity layouts to present the information in the application.

See [Configuring a lifecycle](#).

Activity Flow

Use an activity flow for simple workflow processes.

Activity flows are useful when you need a very simple workflow with a set of tasks with minimal configuration. They are beneficial to use when you would like to make available multiple options to the user to choose a process. For example, if you have a new employee that joins the company and you want to trigger a specific workflow based on the location of the employee. If you design 3 activity flows on an entity, then the user can choose which activity flow to run in the runtime.

Activity flows have the following characteristics:

- The model of an activity flow is a very simple grid, which makes it easy to quickly create a workflow.
- Activity flows complement a lifecycle.
- Activity flows are meant for departmental specific processes that are triggered by users in the application.
- An entity can have multiple activity flows, allowing users to choose at runtime which one to run.
- Multiple activity flows can run successively or concurrently.
- Activity flows can be triggered from a lifecycle.
- Tasks from activity flows and lifecycle share the same context – the entity. If an application provides a task panel, tasks from both activity flows and lifecycle are shown together.
- Activities (tasks) can leverage entity layouts to present the information in the application.

See [Adding an Activity flow](#).

Business Process Model (BPM)

Use a business process model for structured business processes. Business processes typically presume a level of repeatability in the steps taken to accomplish a goal and to complete work. They are often used to automate certain tasks that do not require any human interaction. In your applications, business processes can be used in a variety of different ways.

Business process models have the following characteristics:

- Business processes can automate extremely complex business processes.
- Business processes can use decision tables to model complex business logic (rules).
- Short-lived processes can act like scripts (very efficient) and be triggered from an entity lifecycle.
- A business process can access and update any item that triggered it.
- Business processes can coordinate calls to any service(s).
- Business processes can trigger changes reflected in the user interface in real time.
- Business processes be triggered via an entity rule.

- Activities (tasks) can leverage entity layouts to present the information in the application.

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

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 the application. 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. Application users can select a state in the Inbox to list the activities associated with it.
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  (Lookup a value) to browse and select the related business process or Web service.

Property	Description
	<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 the application, 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 the application, 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 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.

Note: See [Availability of action buttons](#) for more information.

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 expressions](#).

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 expressions.</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 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.

Property	Description
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 the application. 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 the application, it must have primary transition or be in a completed state.</p>
User Event name supports translations	In the application, Lifecycle user event translations are shown in the actions panel as Lifecycle actions.

Lifecycle Progress panel

The Lifecycle Progress panel enables application 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 users you need to add it to a layout. See [Adding item layouts](#).

Activities

By adding activities to a Lifecycle, you can define the tasks that are presented to application users 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 the following types of activities to a Lifecycle model.

- Human activity
- Activity flow
- Business Process activity

Human activity

A human activity defines a task to be executed by a person. The activity can be accessed from the item's Task panel in the application. As part of a human activity, you can configure entity layouts and work assignments (team/work list/role/user). By default, an activity in a Lifecycle model is a human activity. If no layout is linked to an activity, the activity still appears as a task that can be completed in the item's Tasks panel but it 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.

Activity flow

An Activity flow activity defines an activity flow to be executed as part of the lifecycle. When the activity flow is started, a new activity flow instance is created that is linked to the item. The tasks that are created in the activity flow instance are visible in the Tasks panel of the item.

If you make changes to an activity flow invoked at a specific stage in a lifecycle, the changes are effective for all the items in which the processing has not reached that stage. For example, assume that you have an Order Management application that has an activity flow for Approvals and the Approvals activity flow is changed. The changed activity flow is invoked for all orders for which the corresponding lifecycle has not reached approval stage.

Business Process activity

A Business Process activity defines a business process model to be executed as part of the 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 Tasks panel of the item.

If you make changes to a business process or lifecycle model, the changes impact only new items (entity instances) in the system. There is no impact on existing items. For example, assume that you have an Order Management application that has a lifecycle with a certain set of tasks and you have 50 orders in the system. If you make changes to that lifecycle and add three new tasks, the existing 50 orders in the system are not impacted. Only new orders that are created will have the three new tasks that you added to the lifecycle.

Activities configured in the Lifecycle model are referred to as tasks in the application. See [Using My Inbox in the application](#) for information about working with tasks in the application.

- 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 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 application 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.

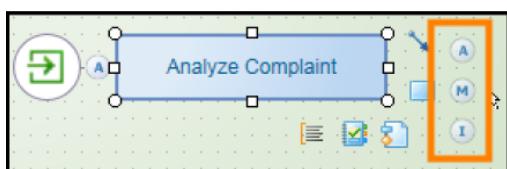
As you hover over the objects to which you can connect, small letters appear next to each object.

After designing the lifecycle model, set the properties of the connectors to define when they should be triggered. This ensures that the lifecycle model executes exactly as you intend 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 it 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 are 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. Linking activities as follow-ups is possible among activities within a state only. Possible connector types are described in the following sections.

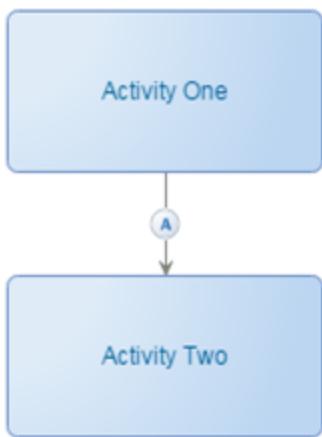
Automatic follow-up connector

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

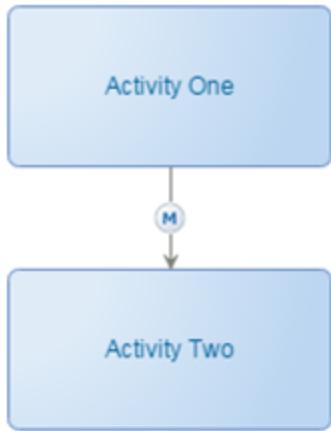
After a state entry, document received event, or business process activity, only automatic (not manual or intermediate) follow-up connectors can be drawn.



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 be drawn only after a human activity. When the human activity is completed from the item's task list, 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.

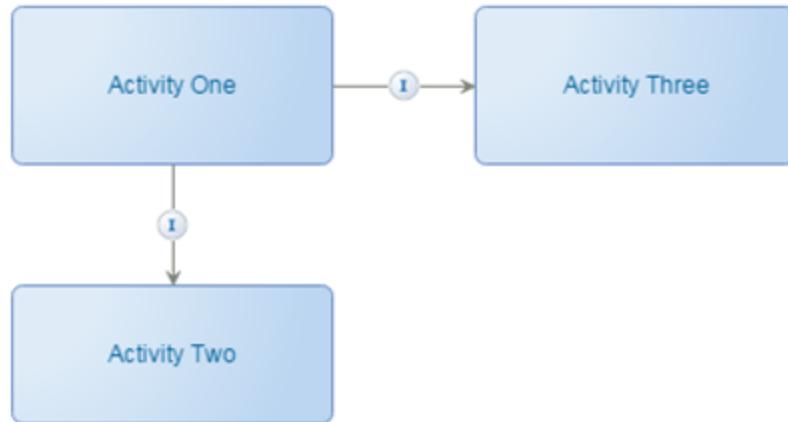


When Activity One and Activity Two are connected through a manual connector, completion of Activity One provides an option to configure and add Activity Two, if relevant.

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 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 the application.

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 are 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 application 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 an entity property, click  (Lookup a value) 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 the lifecycle model level is 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)	Select This activity must be performed (required activity) to ensure that execution of the current activity is enforced. By default, activities of type automatic follow-up are planned during lifecycle execution. However, application users still have an option to remove such activities from execution through the follow-up view displayed on completion of the task. Setting this property disables the removal and forces users to perform the action to complete

Property	Description
	<p>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 the application.

Duration tab

Property	Description
Business Calendar	<p>Enables you to associate the timely business activities with the Lifecycle model. Select the required Business Calendar by clicking  (Lookup) and 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 is applicable for all its activities. You can override the default business calendar at an activity level using this property.</p> <p>See <i>Working with a Business Calendar</i> in the <i>AppWorks Platform Advanced Development</i> 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, specify the duration dynamically by providing an entity property. Click 

Property	Description
	<p>(Lookup) to select the property. See Configuring expressions > Configuring an expression to set a property.</p> <p>By default, the duration configured for the lifecycle model is 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 should 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 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, 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 the *AppWorks Platform Advanced Development* documentation.

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 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. The following options are available: <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. 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.
InstanceId	Opens the expression editor so that you can look for an InstanceId. See Configuring expressions > Configuring an expression to set a property

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.

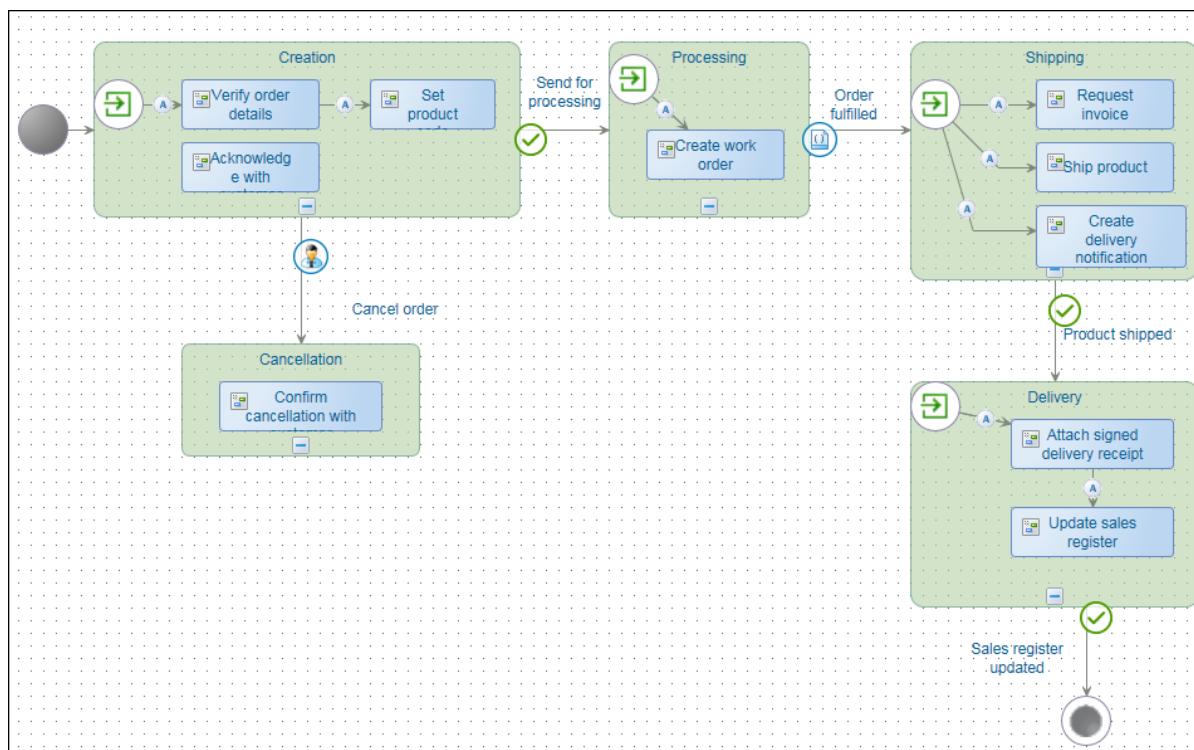
Lifecycle examples

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

Order management

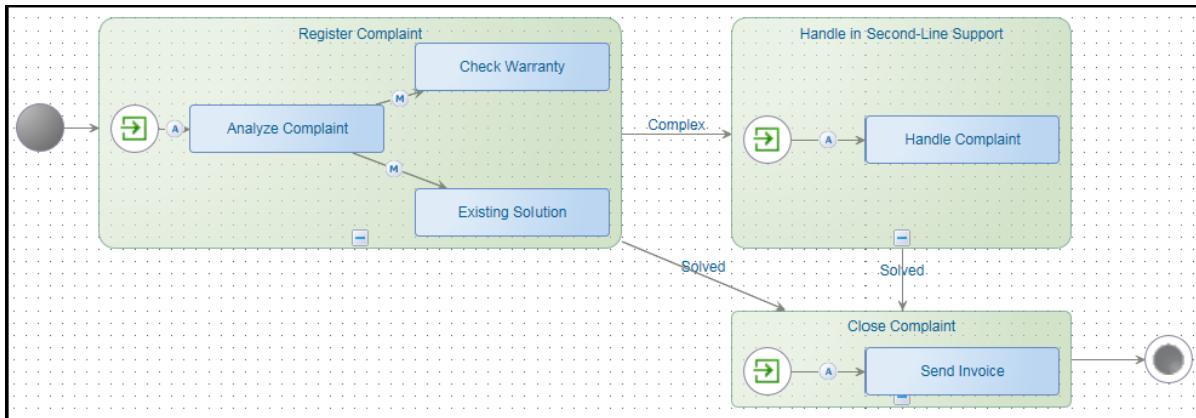
An example of an order management Lifecycle follows.

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 (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. Navigate to Available building blocks > Business logic, and then click **Lifecycle**. The lifecycle designer opens.

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

3. Build your model by dragging the required components to the canvas and configuring their properties. See the following topics:
 - [Lifecycle Designer](#)
 - [Constructs](#)
 - [Configuring the task entity](#)
 - [Configuring expressions](#)
 - [Validating and building the model](#)

4. Define the properties of the lifecycle model. See [Lifecycle model properties](#).
5. Construct the model. See [Creating a Lifecycle model](#).
6. Configure tasks using the Task list building block. See [Configuring the task entity](#).
7. Click  (Save).

Note: You can also access the lifecycle designer from the details pane or the Added building blocks pane.

To work with Low-code design, all service containers interacting with entities, and the WS-Appserver that called the entity runtime must run in TomEE, by configuring them to the Application Server OS process.

For example, for lifecycle these are Business Process Management and Notification service containers.

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

Property display name	Property name	How to access	Description
			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.
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	The ID of the current state. 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

Property display name	Property name	How to access	Description
			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 application users the ability to plan tasks. In addition to the standard actions, all the transition types of user event are exposed as action buttons.

You can restrict access to these actions 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

You can graphically depict how an entity moves through a business process using the lifecycle designer to create a lifecycle model.

Tip: It is a best practice 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:

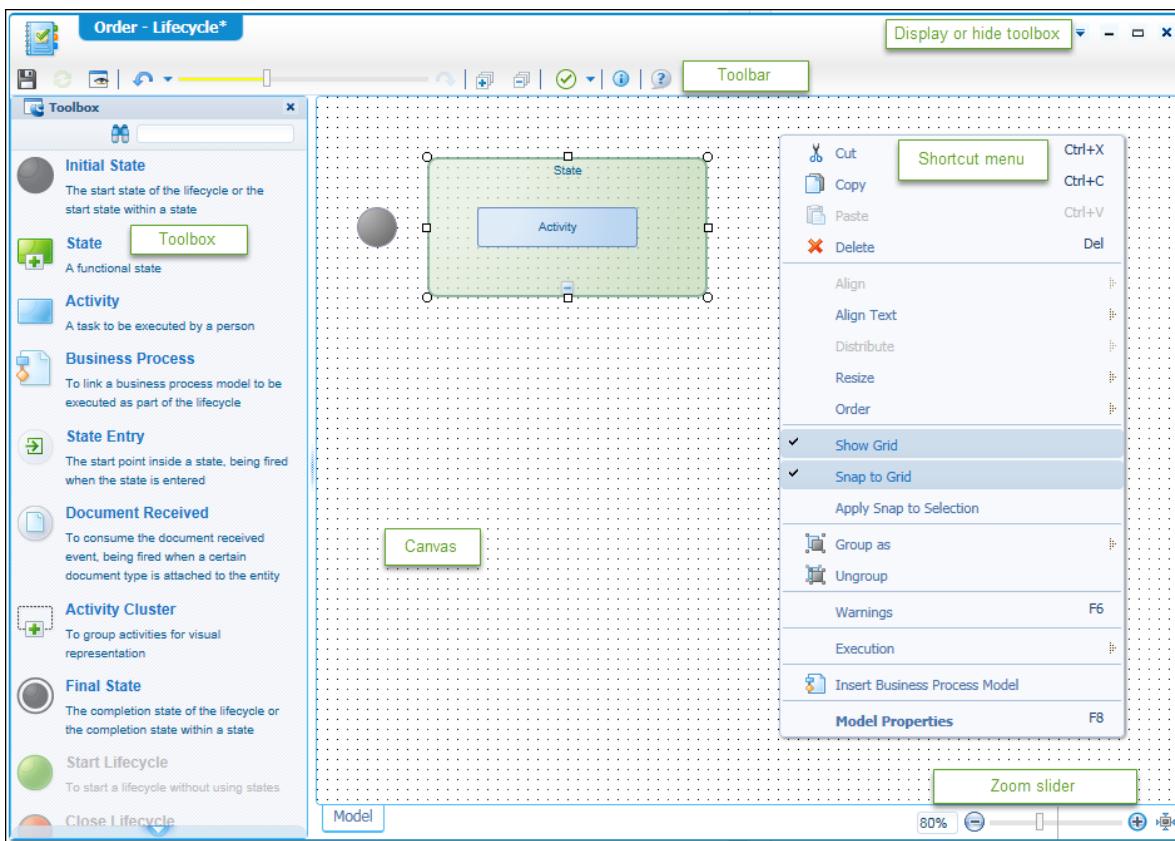
1. In Workspace Documents, open the entity.
2. Navigate to Available building blocks > Business logic, and then click **Lifecycle**. The lifecycle designer opens.
3. Build your model by dragging the required components to the canvas and configuring their properties. See the following topics:
 - [Lifecycle Designer](#)
 - [Constructs](#)
 - [Configuring the task entity](#)
 - [Configuring expressions](#)
 - [Validating and building the model](#)
4. Click  (Save).

Note: You can also access the lifecycle designer from the details pane or the Added building blocks pane.

If you make changes to a business process or lifecycle model, the changes impact only new entity instances in the application. There is no impact to existing entity instances. For example, assume that you have an Order Management application that has a lifecycle with a certain set of tasks and you have 50 orders in the application. If you then make changes to that lifecycle and add three new tasks, the existing 50 order instances in the application are not impacted. Only new orders that are created in the application will have the three new tasks that you added to the lifecycle.

Lifecycle Designer

The Lifecycle Designer is a graphical tool that helps you efficiently build and format a lifecycle.



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 specific 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

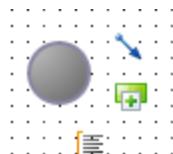
If you right-click the canvas, shortcuts are available to help you with the design process.

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.

Shortcut	Description
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 Process Model	Inserts a business process model that you select in the Lifecycle.
Lifecycle Design Properties	Opens the Properties sheet for the model at the bottom of the window. See Lifecycle model properties .

Tip: The Lifecycle Designer provides a 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	<p>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 application users perform an 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.</p>
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"> Select Business Process or Web Service from the Applicability Service list. Click  (Lookup) 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

Property	Description
	<p>transactions must be performed.</p> <ul style="list-style-type: none"> ■ 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 <i>AppWorks Platform Advanced Development</i> 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 a value for) 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 *AppWorks Platform Advanced Development* 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 States .	State transition State Final State Text Annotation
	Activity	A task to be executed by a person. See Activities .	Connector <ul style="list-style-type: none">▪ Automatic follow-up connector▪ Manual follow-up connector▪ Intermediate follow-up connector Activity Activity Flow <ul style="list-style-type: none">▪ Automatic follow-up connector▪ Manual follow-up connector▪ Intermediate follow-up connector Business Process <ul style="list-style-type: none">▪ Automatic follow-up connector▪ Manual follow-up connector▪ Intermediate follow-up connector 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 Activity Flow Close Lifecycle Text Annotation
	Activity flow	An Activity flow to be executed as part of the Lifecycle.	Automatic follow-up connector Activity Activity Flow Business Process Close Lifecycle

Icon	Construct	Description	Possible elements that can be added from the construct
			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 Activity Flow Text Annotation
	Document Received	The document received event that is triggered when a certain document is attached to the item. 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. In the application, 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.	Automatic follow-up connector Activity Business Process Activity Flow Text Annotation
	Activity Cluster	Groups activities for visual representation.	Text Annotation
	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 Activity Flow

Icon	Construct	Description	Possible elements that can be added from the construct
			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 Task list building block.

To configure the LifecycleTask child entity:

1. In the Added building blocks pane, select **Task list**.
The Task list building block is displayed in the details pane.
2. In Advanced configuration, click **Task list entity configurations**.

Important: By default, the child entity also creates Identity and Relationship building blocks. The Activity flow task building block is also added by default if the parent entity has an Activity flow building block. You must 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, application 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 the application.

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](#) When using these properties in a building block (such as a form), you can assign them labels that are meaningful for your application.

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. You can use both calculated properties and properties shown in the following table in the [expression editor](#).

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	The date when the task is due. 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.
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	Information pertaining to task when it is forwarded to another user: SENDER - The DN of the sender who sent the message initially. RECEIVER- The DN of the receiver who received the message initially. MEMO - The message that was forwarded.
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.
Sender Name	The name of user who sent the task.

Property	Description
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 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 application users.

Important: Do not include a Delete button in the action bar because the delete action will not be visible 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 item 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.

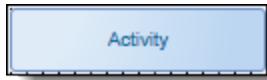
To specify the form to use to work with each activity:

1. Create a different form for each activity.
2. Include each form in a Form panel in a layout.
3. Select the layout in the layout properties.

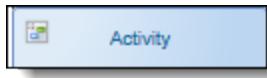
To associate a Lifecycle activity with a layout:

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 application 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 expressions](#).

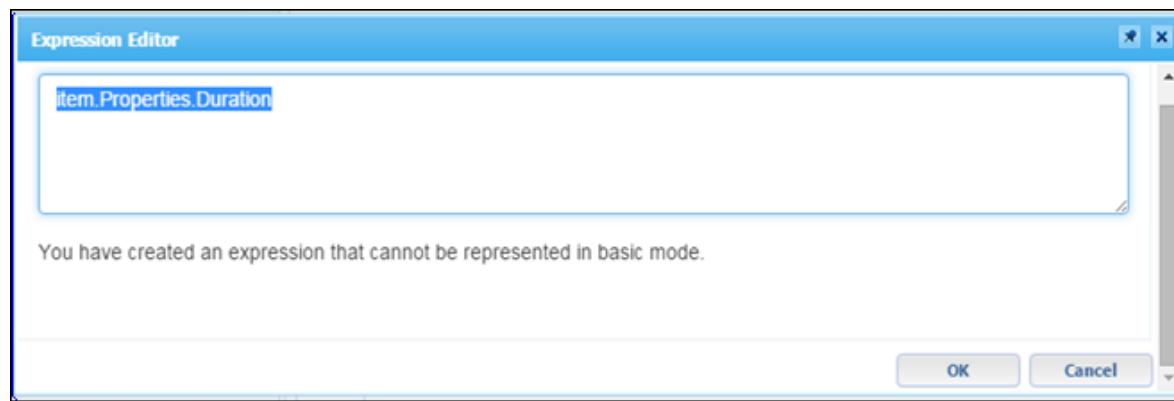
Configuring an expression to set a property

You can use entity properties to define various properties of Lifecycle objects such as duration, assignee, work distribution, and so forth. In the application, 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 > Decimal values](#).

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) For example:

Property name	Expected property value or type
	cn=pjohn, cn=organizational users, o=system, cn=cordys, cn=defaultInst, o=opentext.net
Role	Role distinguished name (DN). For example : cn=Service Agent, cn=organizational roles, o=system, cn=cordys, cn=defaultInst, o=lab.opentext.com
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	A value from 1 to 5 5 - Highest priority 1 - Lowest priority

Validating and building the model

You need to validate and build the model to check for any warnings and errors in the design. The validation process displays a list of error, warning, and informational messages.

When validation is successful you can publish the entity, which also publishes the lifecycle changes. Published changes of the lifecycle model are applicable only for newly created items and not for previously created and published entity items.

Using My Inbox in the application

Every application user is assigned an inbox for sending and receiving tasks. The inbox is configured for a given user profile and functions like a basic mailbox.

Note: You cannot customize My Inbox.

In your application, My Inbox displays tasks that are assigned from the user's AppWorks Platform inbox. Users access My Inbox to view and act on the following lists under the My Inbox category in the application.

- All Tasks - All tasks that are assigned to the user's inbox.
- Personal Tasks - All tasks that are assigned to the user.
- Roles Tasks - All tasks that are assigned to Roles that are assigned to the user.
- Teams Tasks - All tasks that are assigned to Teams the user is a part of.
- WorkList Tasks - All tasks that are sent to worklists the user is assigned to.

These lists do not display tasks if there are no tasks assigned or if the user does not have the required permissions.

The tasks sent to the inbox have several attributes, which are displayed as the following columns in My Inbox under each list:

- Priority
- State
- Subject
- Source Name
- Delivery Date
- StartDate
- DueDate
- Target

The following sections assume that tasks were generated within the application. Users edit them using the application. However, they may also see tasks from the AppWorks Platform Inbox. Those tasks are edited using AppWorks Platform.

Through My Inbox, users can perform the following actions on the tasks in your Inbox:

- Claim - Claim the tasks assigned to their work lists
- Start - Start executing their tasks.
- Pause - Temporarily stop working on a task.
- Delegate - Delegate a task to other users.
- Stop - Completely stop working on a task.
- Complete - Complete a task.
- Skip - Skip a task.
- Modify Start Date - Modify the start date of a task
- Forward - Forward a task to other users.
- Resume - Resume a task that is paused.

For more information on the operations that can be performed on the tasks in the Inbox, see [Working with tasks in application lists](#).

In addition to the above operations, users can also search for specific tasks in the list.

A worklist manager, team lead, or a user with a specific role can do the following:

- Work on the tasks for which they are the users
- Assign a task to a user
- Suspend a task
- Forward a task to another work list or team
- View the status of various tasks belonging to the teams in the list and so on.

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.

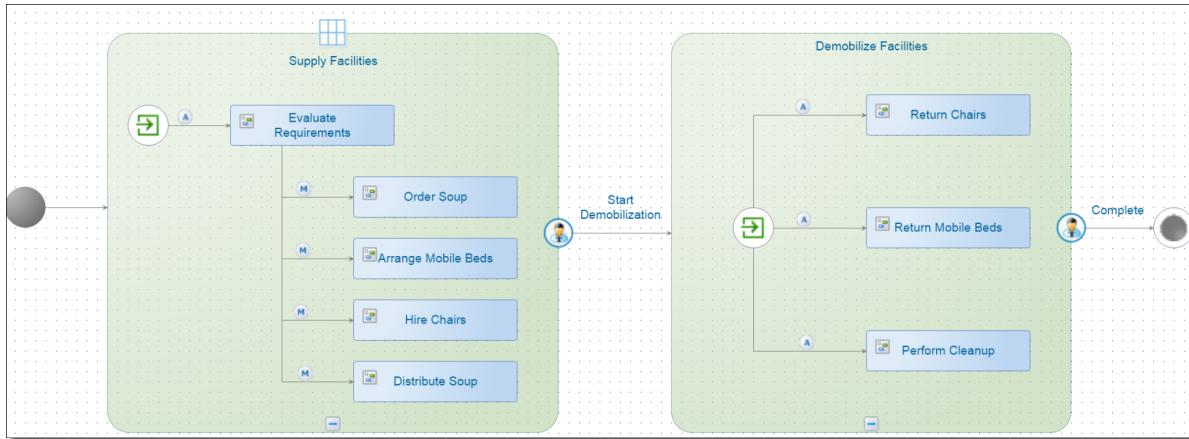
The business logic for this situation is defined based on parameters called Temperature and Duration of the delay. Typically this information is captured during the creation of an item in the application and the data is used in the Lifecycle model. Based on this information, the evaluation of applicability 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 building 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 building 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 the application.
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. See the AppWorks Platform Advanced Development documentation for details.

```
<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:ArrangeMobileBeds>string</ns:ArrangeMobileBeds>
        <ns:HireChairs>string</ns:HireChairs>
    </ns:Activities>
</ns:FacilitiesData>
```

Creating the Applicability Service using a contract

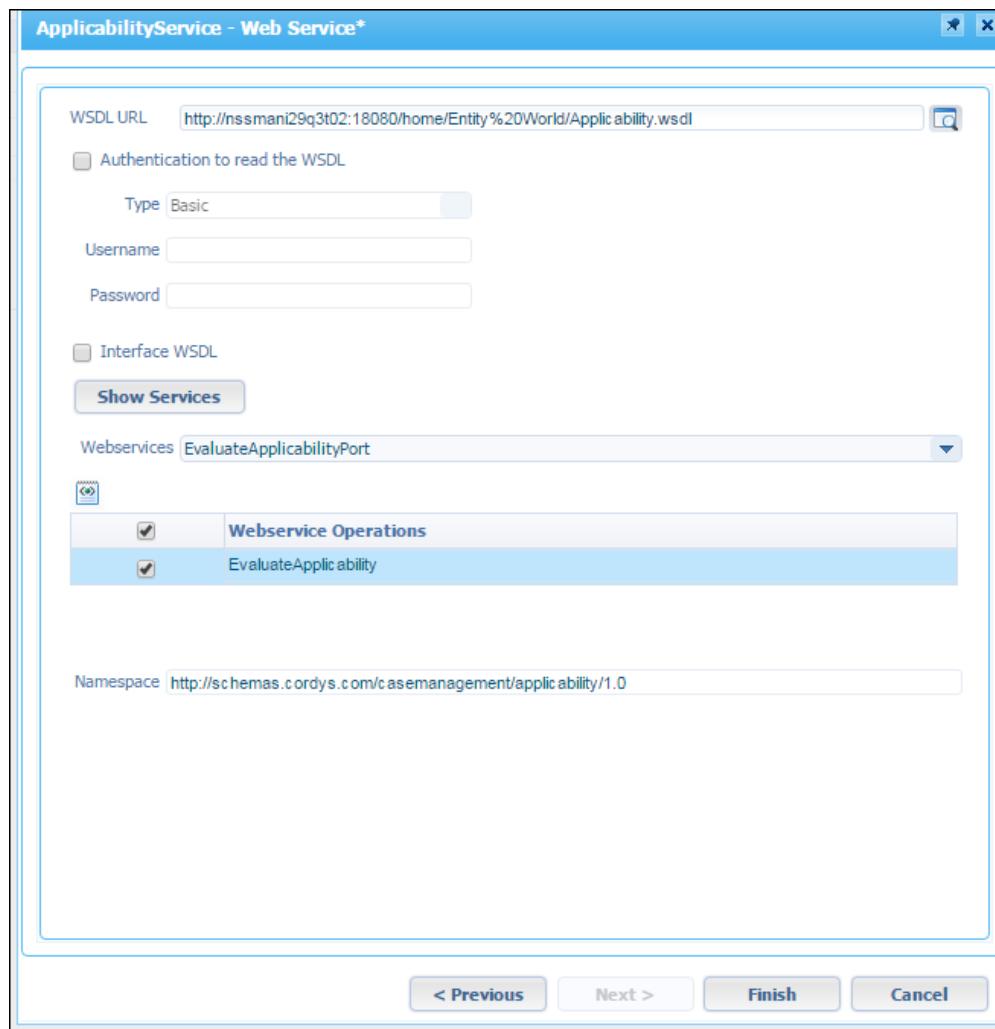
The next step is to create the Applicability Service using a contract.

To create the Applicability Service using a contract:

1. Search for the WSDL contract in the [AppWorks Developer community](#) and copy it to the development server.

For example, if you copy the WSDL file to the following location: <installed folder>\defaultInst\webroot\shared\<newfolder>\ the WSDL URL is <http://<servername>:<port>/home/<newfolder>/EvaluateApplicability.wsdl>.

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.

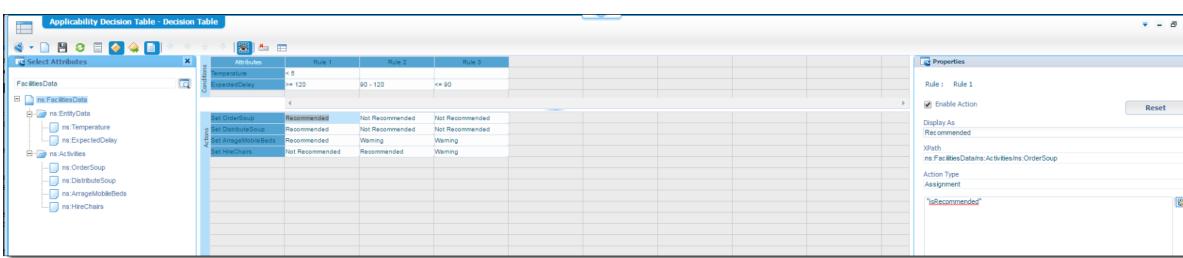
This generates a skeleton of the business process with the defined contract. You need to implement this logic in the business process. For this example, decision logic will be written.

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:

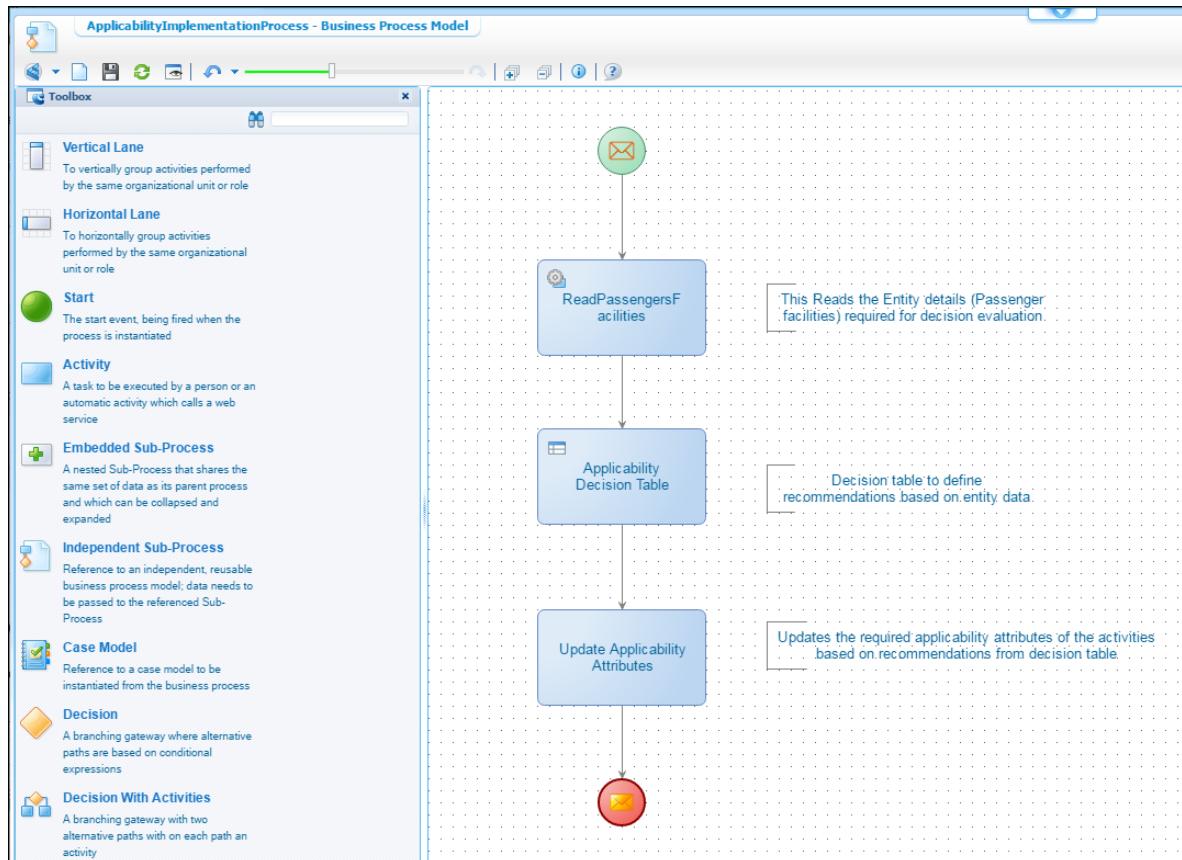


Updating the business process

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.

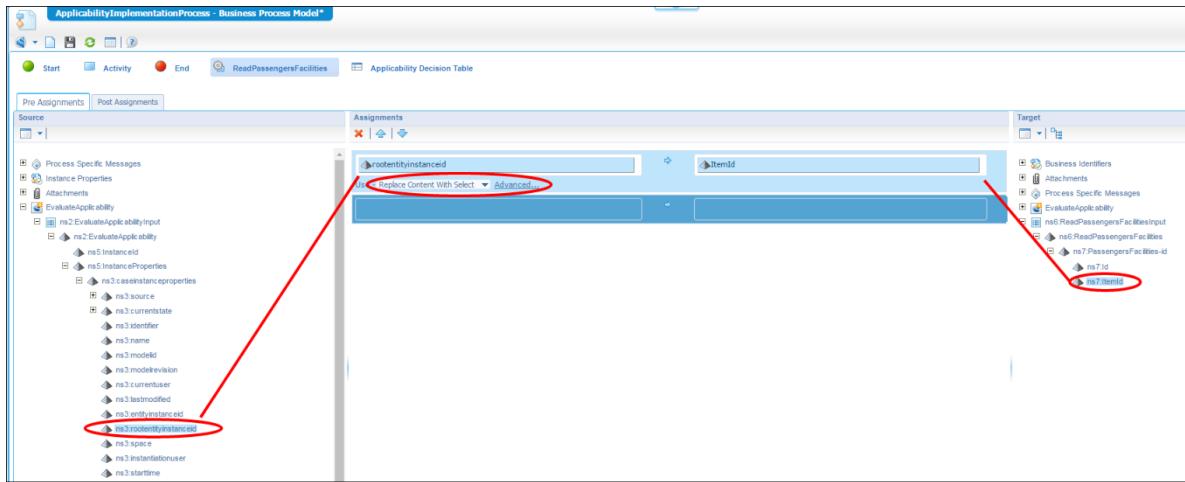


Building the message map assignments

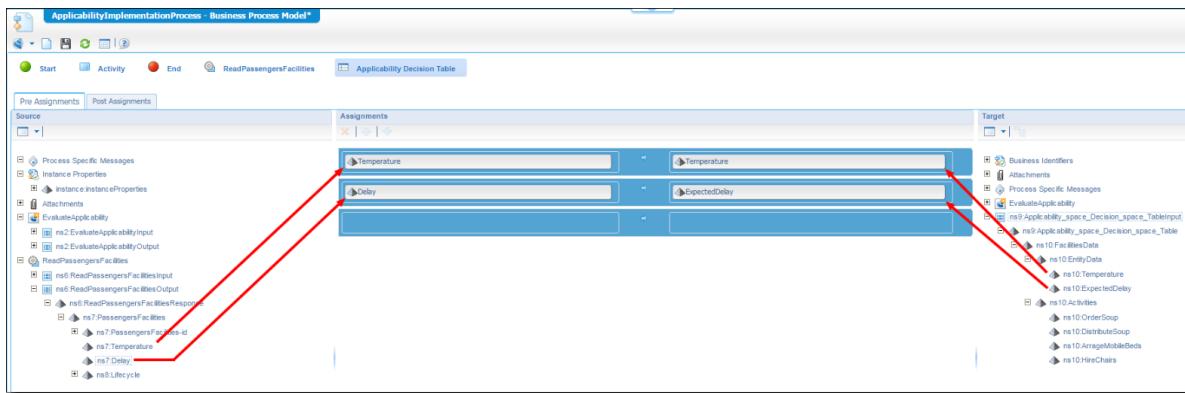
The next step is to build the message map assignments.

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 ReadPassengerFacilitiesOutput 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 ReadPassengerFacilitiesOutput 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.

The figure consists of three screenshots of the Business Process Designer interface, illustrating the steps to update assignments in the consolidated view.

Screenshot 1: Shows the 'Assignments' tab in the consolidated view. A red circle highlights the 'Replace With Select' dropdown menu under the 'Followups' section. The 'Source' pane on the left lists various process components and their properties.

Screenshot 2: Shows the 'Assignments' tab after the update. A red box highlights the updated assignment expressions for 'Followups' in the 'Target' column. The 'Source' pane remains the same.

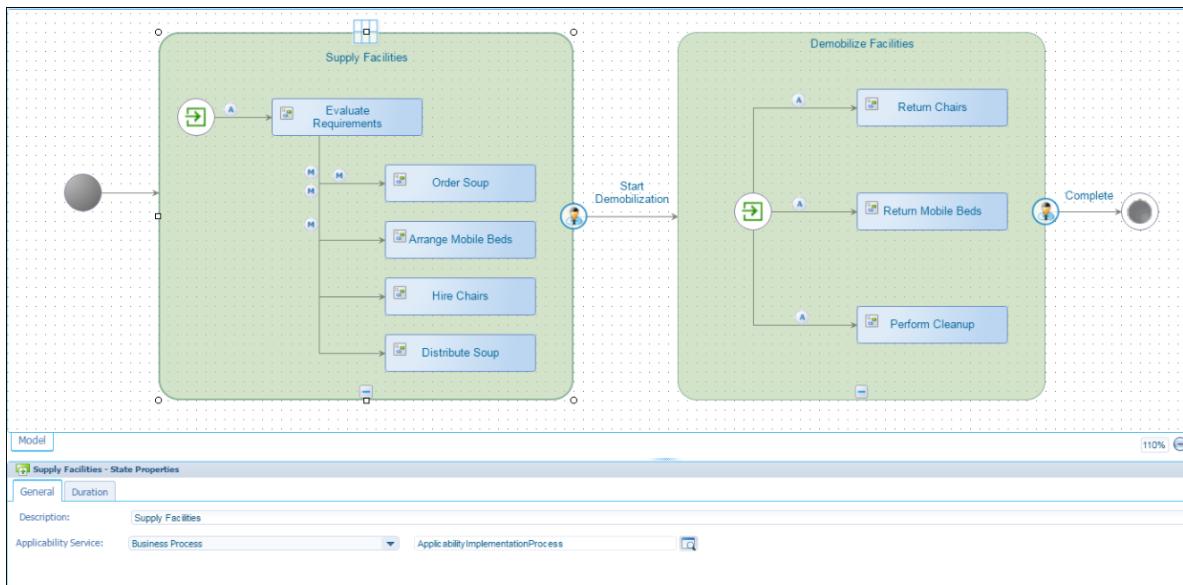
Screenshot 3: Shows the 'Process Flow' tab. A red box highlights the 'Start' node, which contains a 'Update Applicability Attributes' step. This step is part of a larger sequence involving 'ReadPassengersFacilities' and 'Applicability Decision Table' nodes, each with multiple 'Replace Content With Select' steps.

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 the application.

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 the application 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 modal dialog titled "Add follow up tasks". On the left, under "Available tasks", there is a list of four items: "Order Soup" (highlighted with a blue background), "Arrange Mobile Beds", "Hire Chairs", and "Distribute Soup". Each item has a small icon and a thumbs-up or thumbs-down button next to it. On the right, the "Assignment type" is set to "Individual" and assigned to "cordys". The "Due on" section is set to "Duration" (radio button selected) with values 0 Days, 0 Hrs, and 0 Mins. At the bottom right are "OK" and "Cancel" buttons.

Adding an 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. It is based on a grid structure, with each row corresponding to a step in the business process.

Activity flows complement Lifecycle. Activity flows are intended to be departmental specific processes that business users maintain at run time. Activity flows can be initiated as activities in a lifecycle state. This enables you to provide a much broader scope for a lifecycle by including department specific functions as activity flows within the lifecycle. For example, invoice processing might involve verification, approvals, posting, payment, and so forth. Approvals that are specific to each department and maintained by department heads are the candidates for activity flows.

Activity flows offer some benefits over lifecycles:

- Only one lifecycle can be created in an entity. However, multiple activity flows can be created in an entity. Since you can create multiple activity flows, users choose which flow to run at runtime.
- You can run multiple activity flows successively, or even concurrently.
- Activity Flow models are very simple (in a grid). This makes them easy for low-code developers to build without needing a deep understanding of a case model. Lifecycles are more advanced and allow you to add much more advanced functionality.
- Understanding and working with activity flows does not require users to have knowledge about BPMN, case management, and so forth.

Tasks from Activity flows and Lifecycle share the same context (entity). Therefore, if you your application has a Task panel, it shows tasks from both Activity flows and Lifecycle together.

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 application users the ability to trigger activity flows.

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. Navigate to Available building blocks > Business logic, and then click **Activity flow**. The Add activity flow dialog box opens.

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

1. In Label, type the name to show in the application.
2. In Name, the Label value is filled by default.
3. Click **Add**.
The activity flow configuration tab opens.
4. To configure the activity flow, enter the following information:

Field	Description
Activity name	Name of the activity to execute. The name will be visible in the Task panel for the business users.
Previous activity	To sequence the steps in the business process, use this to link the predecessor activity. This enables the business process to determine what must be released when an activity is complete. If this option is left as None, the activity is released as soon as the activity flow is initiated.
Assign to	Assignee for the task. The following options are available: <ul style="list-style-type: none"> ■ Signed-in user: By default, all the activities started by an activity flow are assigned to the signed-in user at runtime. ■ Dynamic user: Select the property containing the user name. ■ Role: Activity assigned to a role defined in the workspace. ■ Dynamic role: Select the property containing the role name. ■ Organizational unit: Activity assigned to the specified organizational unit. ■ Dynamic organizational unit: Select the property containing the organizational unit name.
User interface	Layout that is created on a child task entity for this task.
Due in days	Due date for the task in days.

4. Click  (Save).

Note:

- To make more updates to the configuration, click **Activity flow designer**.
- The activity flow details pane displays information about the available building block actions, relationship with other components, and building block related properties.

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 expressions](#).

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. Navigate to Added building blocks and click the activity flow.
3. To open the designer, click Open designer in the Added building blocks pane or Activity flow designer in the activity flow details pane.
The designer opens.
4. To specify when to automatically initiate the activity flow, click the **Settings** tab and select one of the following options to trigger this activity flow:
 - To initiate the activity flow when a user creates an item, select **When an instance of this entity is created**.
 - To initiate the activity flow when the specified condition is satisfied, select **When a condition is satisfied**.
 - To initiate the activity flow after completion of another activity flow, select **When one of the following activity flows is complete**.

This option provides additional options:

 - Use  (Browse) to select the activity flow.
 - Use + (Plus) and - (Minus) to add or remove the activity flow.

Note: If multiple activity flows are selected, completion of each of these activity flows initiates an instance of this activity flow.
5. To define conditions for the rule, click the Basic tab. See [Using expressions](#).
6. Click  (Save).

Integrating BPM processes and entities

The business objectives of an enterprise determine the way its business processes are modeled. Using AppWorks Platform, you can analyze the existing processes and identify scope for improvement, and design or model processes based on your business goals and objectives. You can develop and deploy the processes, and subsequently monitor what you have deployed. AppWorks Platform enables you to not only design and automate routine tasks, but also to orchestrate and optimize sophisticated processes involving people, information, applications and complex exception-handling scenarios. At the core of the AppWorks Platform solution is a continuous process management cycle that follows the natural flow of an enterprise, involving analysis, design, development, and monitoring.

You typically will include one or more business processes in your low-code applications. 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.

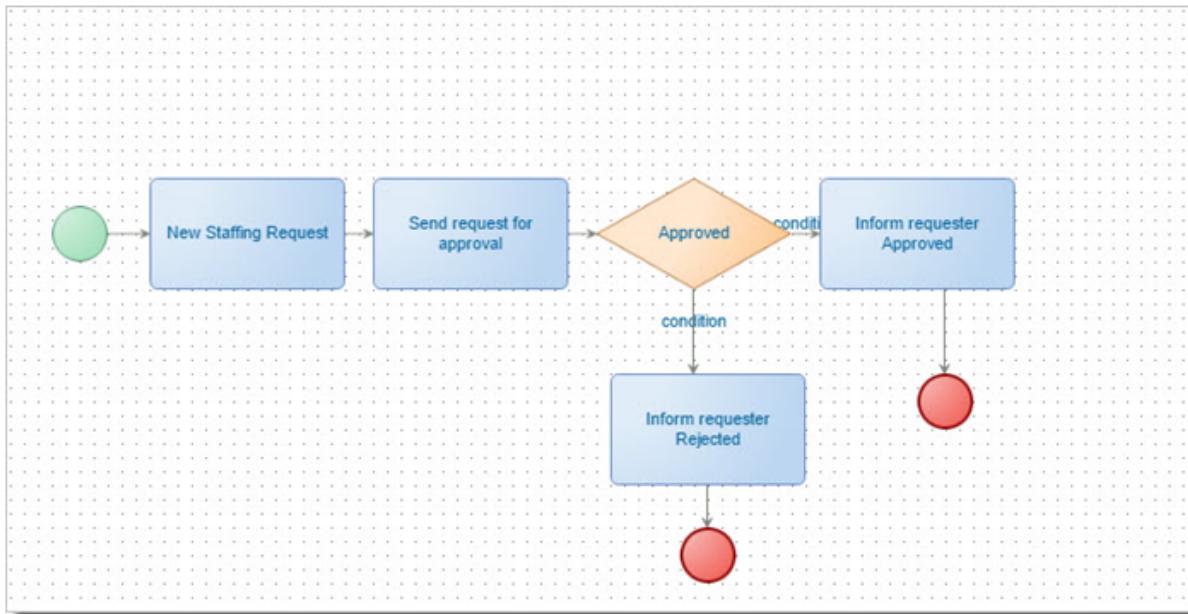
In most cases, the BPM process that triggers from an entity also needs more entity data. Therefore, it is quite common to have a combined approach.

The basic steps are as follows:

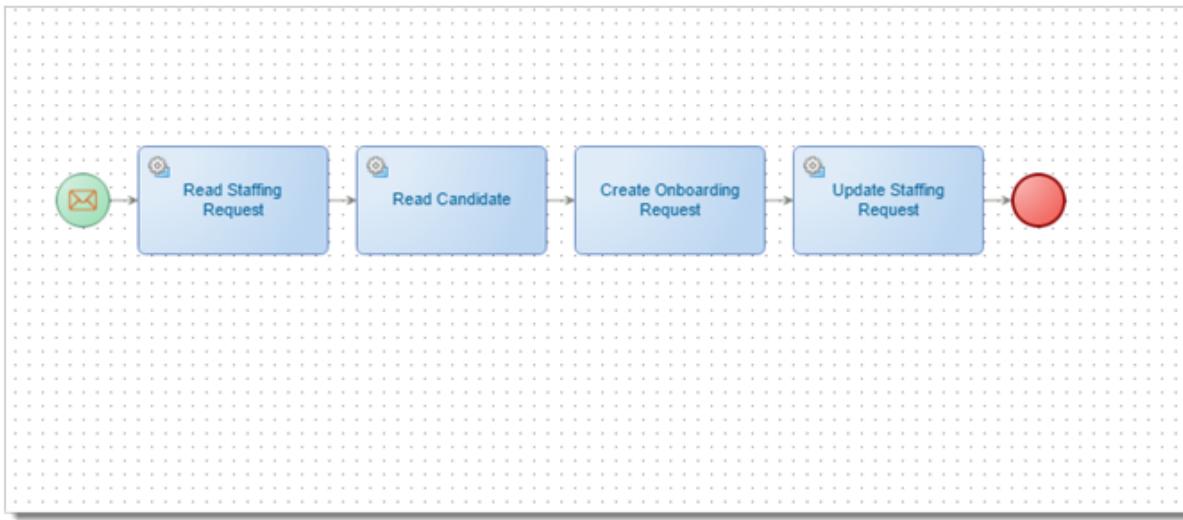
1. Generate web service operations on an entity.
2. Create the BPM process.
3. Add the necessary entity web services as activities in the BPM process.
4. Map the input and output of each activity in the message map.
5. Create a rule on the entity and select the BPM process that you want to trigger.

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.

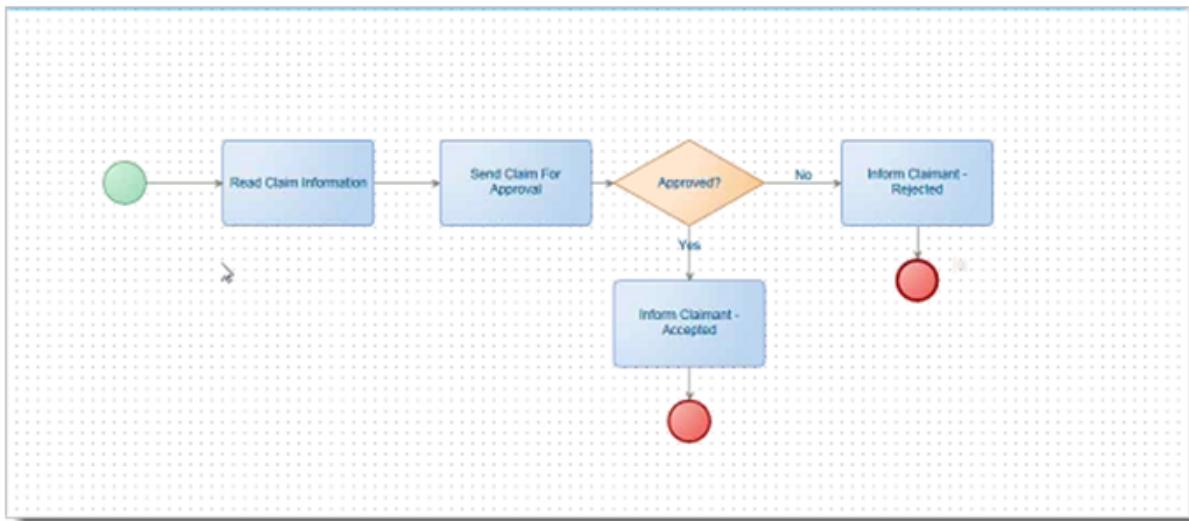


Process overview

1. Create a new workspace and project where you plan to build an entity-based application.
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 the application.

Creating the BPM process model

When you build an entity-based application, 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.

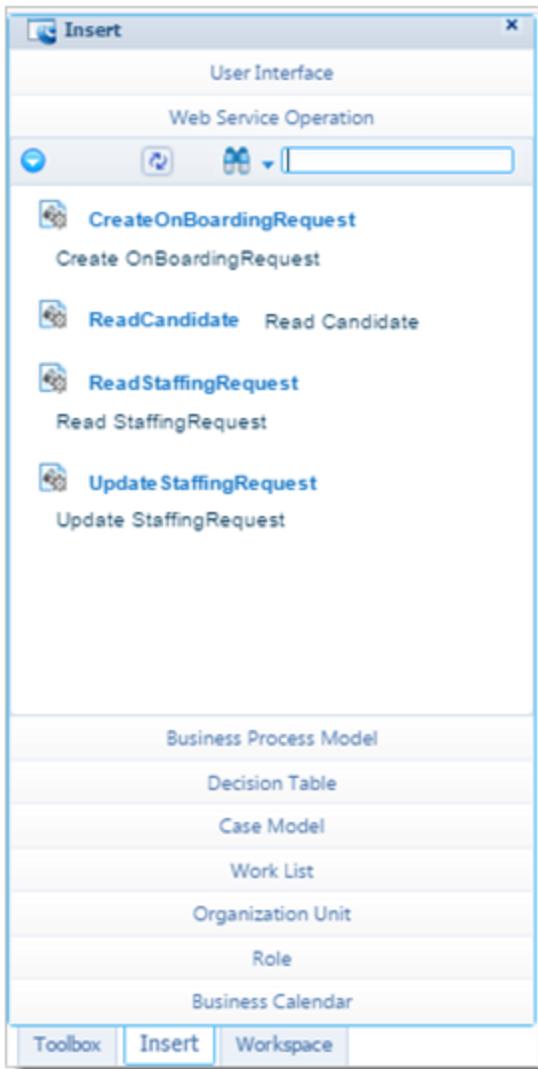


You can integrate your BPM process with your entity application in a variety of ways. For example, you can create a multi-level approval process. Creating a business process is easy but, if you need help, see *Creating a Business Process Model* in the *AppWorks Platform Advanced Development* documentation.

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.

The screenshot shows the 'Rule Properties' dialog box. Under 'When:', 'Action' is selected. Under 'If:', 'Basic' is selected. The condition 'all' of the following are true is set with 'Status' equal to 'Offer Accepted'. Under 'Then:', 'Start Process' is selected. The process 'EmployeeOnboarding' is chosen from the dropdown. A 'Save Process Id' section with a 'Select' dropdown is also visible.

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**.

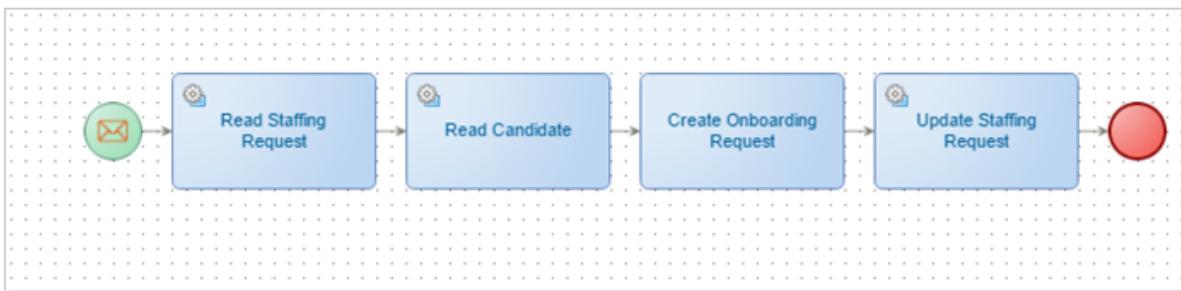
3. Select the basic operations. For example, in a staffing application, after a new candidate has accepted the job, you could initiate an employee onboarding process. HR staff members need to be able to view information to determine where things are in the process. In this case, the Read basic operation 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'

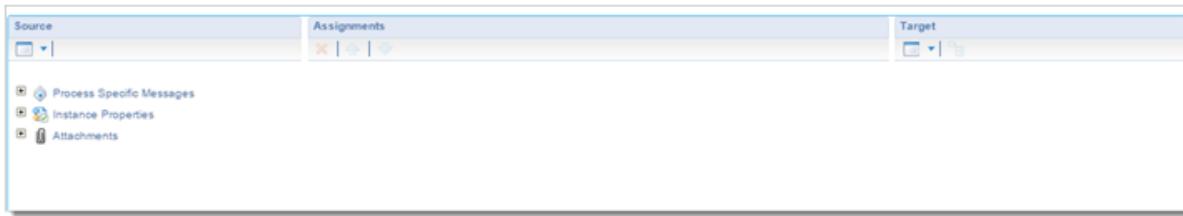
1. Open the process. The Toolbox automatically opens on the left.
2. Click the Insert tab.
3. Open the Web Service Operation section.
4. Drag the web service from the workspace area to the activity in the process.
5. Edit the description.
6. Repeat this process as necessary. The following example shows 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.

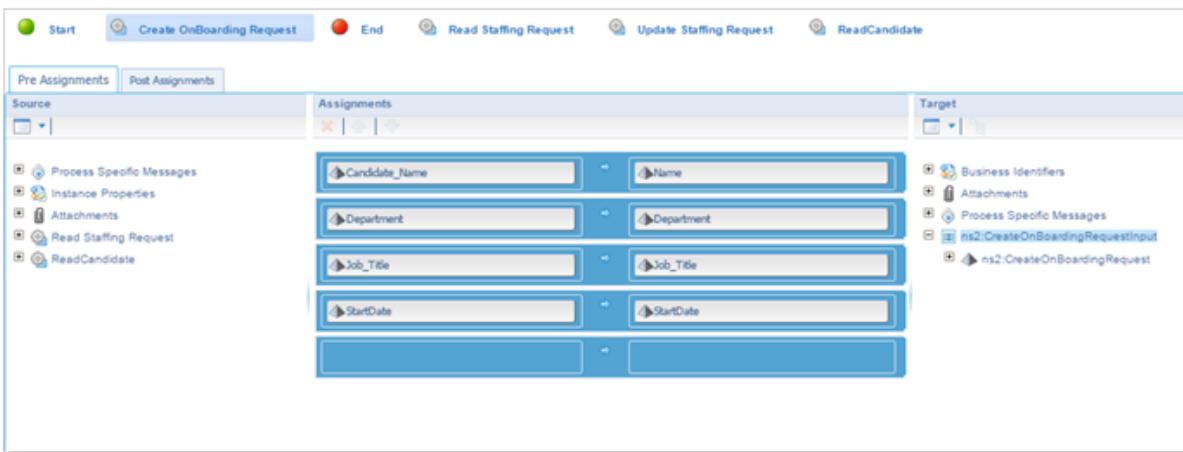


The message map shows three panes: Source, Assignments, and Targets.



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.

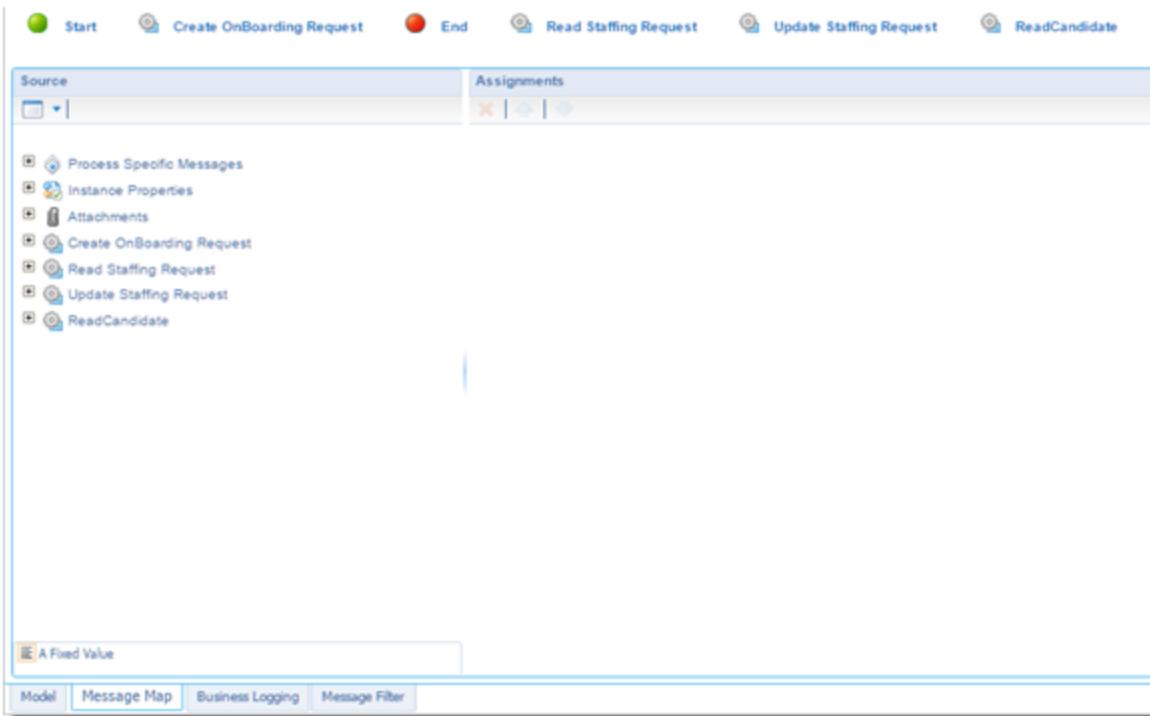


For specific information on creating assignments, see the *AppWorks Platform* documentation.

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 *AppWorks Platform* documentation.

Also see [Creating a business process for a dynamic enumeration](#).

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 AppWorks Platform.

Adding rules

Defining rules that add business logic to an entity is an important part of application development. A rule can be of two types: Action and Event. While an Event rule is used to configure an action that must be triggered for an event, the Action rule is used to create an action that is performed when the Action button is clicked.

Note: Once the Action rule type is added to the entity, you cannot change the rule type to Event. Also, an Event rule cannot be changed to an Action rule.

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

Note: Solution variables cannot be used in rules. Rules that use the user context convert user to User during validation. The only exception to this is for URLs that can be configured when the rule of type **Action** is selected.

To add a rule to an entity:

1. In Workspace Documents, open the entity.
2. Navigate to Available building blocks > Business logic and then select **Rule**.
The Add rule dialog box opens.
3. From the **Type** list, select Action or Event and do the following:

Note: Once the Action rule type is added to the entity, you cannot change the rule type to Event.

- If you select **Event**, provide a **Name**.
- If you select **Action**, provide the **Label on the action button** and **Name**.

Click **Add** to open the Rule designer.

Note: When you click the **Name** box, the label entered is copied here without spaces or invalid characters.

4. On the **Basic** tab or **Advanced** tab, define the conditions for the rule. See [Using expressions](#).
5. In the **Then** area, select an action to perform when the conditions of the rule are met: Show error, Show warning, Show information, Set property, Start process, Hide, Disable, 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 property, select the target and source.
 - For **Start process**, select a process and expand the **more options** link.
 - For Disable or Hide, select a property or category.
 - 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 the application (in forms and lists). See [Apply styles](#) for more information.

6. Click  (Save).

Events that can trigger a rule

When you configure a rule, you need to select the event that will trigger its execution. The following sections describe each event.

An item 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.

The initialized event is not evaluated on the client side and therefore does not apply to Show warning, Disable, and Hide rule actions. On the Show error, Set property, and Start process rules, actions are available when the Action type is selected.

An item 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.

The created event is not evaluated on the client side and therefore does not apply to Show warning, Disable, and Hide rule actions. On the Show error, Set property, and Start process rule, actions are available when the Action type is selected.

For information about how this type of rule is triggered for EIS entities, see [Building the application](#).

A property changes

The rule is triggered when a form is first loaded, and any time a property that is referenced in the rule condition 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 parent and to child relationships are not supported.

An item is deleted

The rule is triggered when an object instance is about to be deleted.

The deleted event is not evaluated on the client side and therefore does not apply to Show warning, Disable, and Hide rule actions. On the Show error, Set property, and Start process rules, actions are available when the action type is selected.

Rule actions

When you configure a rule, you specify the actions that should occur when the conditions of the rule are met. The following sections describe each rule action in detail.

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 [Events that can trigger a rule](#). 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.

On the server, the Show error rule action reports the error message and aborts all the changes that are being made with the specified event. For example, if a rule is defined to be triggered when "An item is created" event occurs and the server detects a rule is in error, the error message is reported to the user and the creation of the instance is prevented. Rules that are defined to be triggered based upon the following events are executed only on the server:

- An item is initialized
- An item is created

- An item is deleted
- A relationship changes

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 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 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 property

A Set property 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 property 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 property rules are also run in the client software to provide responsive feedback, even though they are executed again when the changes are persisted to the server.

In forms, the target property of a Set property rule is automatically disabled because its value is being set by the rule and cannot be changed by a user. If the Set property 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 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.

Start process

Important: 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 an AppWorks 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 a business process by clicking the browse 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.

The following options are available when the **On property change** option is selected:

- The **Execute this rule only when the focus moves away from the entire form** 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 the process only when the specified condition changes from false to true** 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 following option is always enabled:

The **Store the generated process instance ID in a property** functionality enables you to store the process instance ID in a text property. When a process is successfully started, a process instance ID is returned. Optionally, the Start process rule action can store this process instance ID in a text property.

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.

The Save 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 the entity properties that have the right data type (Text) 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.

Hide

In the application, hides the selected property or the form components tagged with a specific category when the rule evaluates to true.

Disable

In the application, disables the selected property or the form components tagged with a specific category when the rule evaluates to true.

Apply styles

In the application, 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 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.

Advanced configurations

The Action rule is executed 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 bar presentation. These types of rules have some advanced options that can be set in the **Advanced configuration** section.

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.

Note: See [Availability of action buttons](#) for more information.

The **Before the action is performed, launch a modal dialog** option provides the ability to display a form to the user before the action is executed or to display a modal dialog box that displays a web page to the user. By default, no dialog box is displayed to the user before the action executed.

A modal dialog box forces a user to interact with it before the user can go back to using the parent application.

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 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. If the Action form contains only components that are read only or not displayed for multiple selection, the Action rule is not displayed.

If you want to display a web page in a modal dialog box before the action is invoked, enter a valid URL in the input box. There are security restrictions that prevent a URL's web pages that are outside the current application TomEE environment from being displayed (cross site restrictions). The URL can contain embedded references to properties that perform a property value substitution to generate appropriate URLs.

The **This action must be signed by the user** option forces the user to re-authenticate with the same credentials as the user is already logged in with every time the action button is clicked. This enables the application to force the user to confirm the user's identity before the action is executed. With this option, a signed user action can be configured as required by some regulations. You may also need to enable the **When the action is performed, write to history log** option. Authentication failures are logged in the OTDS log file, not the entity runtime history file. See the Configuring OTDS for signed user action topic in the *AppWorks Platform Administration Guide*.

The **When the action is performed, write to history log** option is used to log a message to the history log when an action is executed. Optionally, a message can be appended to the standard message by providing text in the input box. The message can use variable substitution to embed the property values that are displayed in the action form by including them in the message. An example follows:

```
Value of TestPropertyA is: {item.Properties.TestPropertyA}
```

The **After the action is performed, redirect the user to a specific location** option provides the ability to automatically navigate to a different page after the action is performed. By default, no navigation is performed after the action is executed.

- **Close current item** closes the current item and navigates to the previous item in the breadcrumb panel. If no 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. Property value substitution can be embedded in the supplied URL so that an appropriate URL is generated before the navigation occurs. Property value(s) can be added in the URL by enclosing the full property name in curly braces. Properties from related entities cannot be used in the property value substitution. Some examples follow.

```
http://localhost/docs/setup.html?{User.Properties.FullName}
```

```
http://localhost/docs/setup.html?{item.Properties.TestPropertyName}
```

```
http://localhost/docs/setup.html?{system.baseURL}
```

Example of 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 Status values of None, Active, and Inactive you might create a rule that enables a user to change a vendor status from None to Active. For this type of rule, the Label 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. In Workspace Documents, open the entity.
2. Navigate to Available building blocks > Business logic and then select **Rule**.
The Add rule dialog box opens.
3. Create a rule and from the **Type** list select if it is an **Event** rule or an **Action** rule.
4. In the **If** area, select **Status > equal to > None**.
5. In the **Then** list, select **Set property**.
6. In **Target**, select **Status**.
7. In **Source**, type "**Active**". Ensure that you include the quotation marks.
8. Save the rule.

You can specify a rule condition by creating an expression using the Expression editor. See [Using expressions](#).

Availability of action buttons

If the **Allow this action when there are errors** check box is used, the action buttons can be enabled or disabled.

This topic describes the availability of action buttons when three types of errors occur: hard error, mandatory property error, and soft error.

Note: Items can be saved and some actions can be performed even when the state of the instance is in error.

The following table describes the different types of errors and the resulting behavior of the action buttons:

Classification of error	When does it occur	Action button availability
Hard	Information entered cannot	All the action buttons are

Classification of error	When does it occur	Action button availability
	be saved, for example, if a letter is entered for an integer property.	disabled. When this happens in a dialog box, the OK button is disabled.
Missing mandatory properties	The required property is visible and empty.	The action button is enabled if the Allow this action when there are errors check box is selected. When this happens in a dialog box, the OK button is disabled.
Soft	A rule of the When a property changes event that displays an error message on a certain condition.	The action button is only enabled if the Allow this action when there are errors check box is selected. When this happens in a dialog box, the OK button is enabled.

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 AppWorks 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 AppWorks Platform.

Note: For the generic details of the web services see [Entity web service details](#).

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 AppWorks Platform not to generate errors if such characters result in invalid URIs. You can do this in the AppWorks 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 *AppWorks 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. Navigate to Available building blocks > Functional, and then click **Web service**. The building block is added to the entity. The namespace of the web service and the available operations are listed in Configuration 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 are used, the package name must not be changed. This would change the namespace and require an update of the places where they are used.

3. In Configuration or Advanced configuration, select the operations to expose to the entity.
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. In the Added building blocks pane, click **Web service**.
The operations are listed in the web service details pane.
3. In Configuration or Advanced configuration, clear the operation 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 all operations are not cleared, you are notified to clear them first.

To create and expose a Find web service operation:

1. In Workspace Documents, open the entity.
2. In the Added building blocks pane, click **Web service**.
The operations are listed in the web service details pane.
3. In Advanced configuration, on the toolbar of the Find operations table, click  (Add).
A row is added the table.
4. 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 AppWorks Platform.

4. On the toolbar or in the row, click  (Open designer).
The designer 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 is detected when you try to validate or publish the entity.

You can define a Find web service operation. However, it is not guaranteed that it will perform on a database that contains a large number of entity instances. If performance deteriorates, alert your database administrator. They can take measures to restore performance.

To update a Find web service operation:

1. In Workspace Documents, open the entity.
2. In the Added building blocks pane, click **Web service**.
The operations are listed in the web service details pane.
3. In Advanced configuration, in the Find operations table, select an operation to update, and then click  (Open designer) on the toolbar or in the row. Alternatively, pause on the row of the operation and click  (Open designer) for the row.
The designer opens.
4. 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. In the Added building blocks pane, click **Web service**.
The operations are listed in the web service details pane.
3. In Advanced configuration, in the Find operations table, select the row of the operation to remove, and then click  (Delete) on the toolbar or in the row. Alternatively, pause on the row of the operation and click  (Delete).
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 any operation is not cleared, you are prompted to clear it.

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 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"
        xmlns:wstxns3="http://schemas/OpenTextOrderManagement/BusinessPartner">
        <Id xmlns="http://schemas/OpenTextOrderManagement/BusinessPartner">1</Id>
      </wstxns3:BusinessPartner-id>
    </Customer>
  </wstxns2:Order>
</wstxns1:CreateOrderResponse>
```

```

<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: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>

```

The Id, together with the namespace, is a unique identifier to identify a single entity item in a database table. Id is the preferred way to retrieve the single entity item.

The Id1..Id2..IdX and ItemId1..ItemId2..ItemIdX are described in [Web services on relationships](#) which is provided to differentiate between parent and (recursive) child-items.

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
                xmlns:ns1="http://schemas/OpenTextOrderManagement/BusinessPartner">
                <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 AppWorks Platform product documentation.

To access the documentation, click  (Help) on the shortcut bar of the AppWorks 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>
```

```
</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 made bidirectional 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 made unidirectional, or is made bidirectional 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">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>
    <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 made bidirectional 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 made unidirectional, or is made bidirectional 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"
  xmlns: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 expressions](#)

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.

If you use parameters in your Find web service, the data type of a parameter must match the data type of the property as shown in the following table.

Property data type	Parameter data type
Big integer	Integer
Boolean	Boolean, Text
Currency	Decimal, Float, Integer
Date	Date, DateTime
Date and time	DateTime, Date

Property data type	Parameter data type
Decimal	Decimal, Float, Integer
Double	Decimal, Float, Integer
Enumerated integer	<i>Not supported</i>
Enumerated text	<i>Not supported</i>
Float	Float, Decimal, Integer
Image	<i>Not supported</i>
Integer	Integer
Text	Text, Decimal, Integer, Float
Long text	Text, Decimal, Integer, Float

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>
```

Caution: An alternative for Id is ItemId. However, it is strongly recommended that you do not use it because it is intended for internal use only. It consists of a GUID that is linked to the entity of the specified Id and the Id itself, separated by a period (for example, <ns0:ItemId>F8B156E1037111E6E9CB0FBF3334FBBF.4</ns0:ItemId>).

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>
```

Note: Date values must be specified in UTC. They cannot be specified in other time zones. A Date value without the trailing Z is assumed to be in UTC zero.

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>
```

Note: Date and Time values must be specified in UTC. They cannot be specified in other time zones. A Date and Time value without the trailing Z is assumed to be in UTC zero.

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 the application

This section provides information about implemented SOAP web services. The functionality of the SOAP web services is based on the application.

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 the application and attach it to a specific 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>
      </Email>
    </sendEmail>
  </SOAP: Body>
</SOAP:Envelope>
```

```

<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
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 the application and attach it to a specific 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 specific 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 specific 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

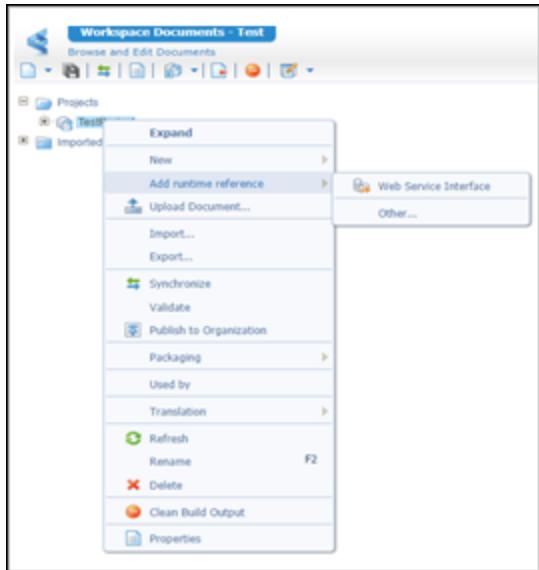
Parameter	Description	Data Type	Required
ItemId	ItemId of the item	String	Yes

Using Email web services in other models

To make the Email web services available in other models (such as BPM or Lifecycle) you need to perform certain steps.

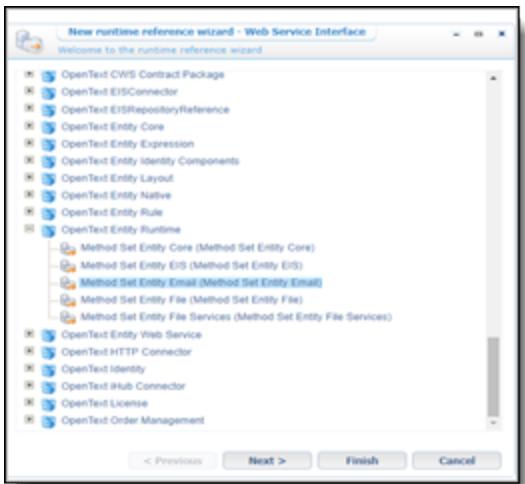
To make the Email web services available in other models:

1. Right-click the project you are working with and select **Add Runtime Reference** > **other**.



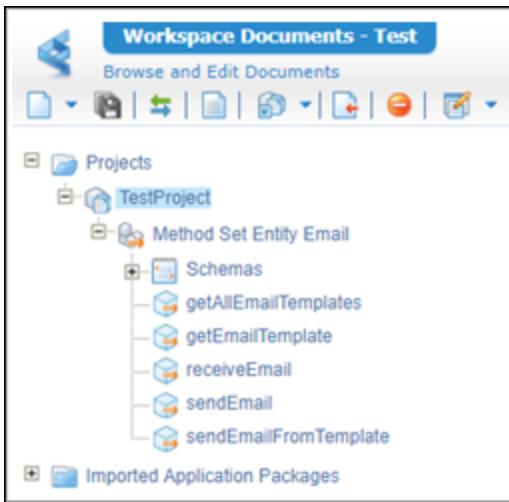
2. Select **Web Service Interface** from the list that is displayed (sometimes **Web Service Interface** may be shown directly when you click Add Runtime Reference).

The list of Runtime References Available is shown.



3. Expand **OpenText Entity Runtime**, select **Method Set Entity Email (Method Set Entity Email)**, and then click **Finish**.

Method Set Entity Email is included in the project. Expanding it shows the list of Email web services available for you to use.



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 the application:

- 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
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>
```

DeleteDocument

The DeleteDocument web service deletes the document from the repository and removes the document details from the entity instance. This is enabled for documents uploaded using the File building block and Content building block.

Note: For documents linked or referenced to the Content building block, this service removes the link from the entity instance. It does not delete the document from the repository.

SOAP request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <DeleteDocument
      xmlns="http://schemas.opentext.com/bps/entity/buildingblock/file">
      <ns0:ItemId
        xmlns:ns0="http://schemas.opentext.com/bps/entity/core">PARAMETER</ns0:ItemId>
      <DeleteItem>false</DeleteItem>
    </DeleteDocument>
  </SOAP:Body>
</SOAP:Envelope>
```

Request parameters

Parameter	Description	Data Type
ItemId	ItemId of the item	String
DeleteItem	Optional parameter that specifies if an	Boolean

Parameter	Description	Data Type
	item must be deleted along with the document	

SOAP response

```
<data>
<ns2:DeleteDocumentResponse
  xmlns:ns2="http://schemas.opentext.com/bps/entity/buildingblock/file"
  xmlns="http://schemas.opentext.com/bps/entity/core">
  <ItemId xmlns="http://schemas.opentext.com/bps/entity/core"
    xmlns:ns2="http://schemas.opentext.com/bps/entity/buildingblock/file">005056C0000
  8A1E7B7F75A7555D55BFF.147458.164675593</ItemId>
</ns2:DeleteDocumentResponse>
</data>
```

Examples

Following is a sample payload for deleting a document that is uploaded to a File building block:

Request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <DeleteDocument
      xmlns="http://schemas.opentext.com/bps/entity/buildingblock/file">
      <ns0:ItemId
        xmlns:ns0="http://schemas.opentext.com/bps/entity/core">F8B156B63B8FA1E88B0A93EB0
      29124C6.16385</ns0:ItemId>
      <DeleteItem>false</DeleteItem>
    </DeleteDocument>
  </SOAP:Body>
</SOAP:Envelope>
```

Response

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <ns2:DeleteDocumentResponse
      xmlns:ns2="http://schemas.opentext.com/bps/entity/buildingblock/file"
      xmlns="http://schemas.opentext.com/bps/entity/core">
```

```

<ItemId xmlns="http://schemas.opentext.com/bps/entity/core"
xmlns:ns2="http://schemas.opentext.com/bps/entity/buildingblock/file">F8B156B63B8
FA1E88B0A93EB029124C6.16385</ItemId>
</ns2:DeleteDocumentResponse>
</SOAP:Body>
</SOAP:Envelope>

```

Following is a sample payload for deleting a document that is uploaded to a Content building block:

Request

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Body>
<DeleteDocument
xmlns="http://schemas.opentext.com/bps/entity/buildingblock/file">
<ns0:ItemId
xmlns:ns0="http://schemas.opentext.com/bps/entity/core">005056C00008A1E7B7F75A755
5D55BFF.147458.164741123</ns0:ItemId>
<DeleteItem>false</DeleteItem>
</DeleteDocument>
</SOAP:Body>
</SOAP:Envelope>

```

Response

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Body>
<ns2:DeleteDocumentResponse
xmlns:ns2="http://schemas.opentext.com/bps/entity/buildingblock/file"
xmlns="http://schemas.opentext.com/bps/entity/core">
<ItemId xmlns="http://schemas.opentext.com/bps/entity/core"
xmlns:ns2="http://schemas.opentext.com/bps/entity/buildingblock/file">005056C0000
8A1E7B7F75A7555D55BFF.147458.164741123</ItemId>
</ns2:DeleteDocumentResponse>
</SOAP:Body>
</SOAP:Envelope>

```

Lifecycle web services

The ExecuteLifecycleAction web service operation enables the user to send a Lifecycle user event action to a specified entity instance even when the Lifecycle instance is versioned in

the current entity. If the operation is successful, the response provides details of the result. If the entity instance does not accept this operation, the response specifies the reason.

Before executing this web service, you need to enable the -

Dcom.opentext.lifecycle.action.execution system property on TomEE. Enabling this property on the TomEE service makes the web service executable on the corresponding platform.

To enable the -Dcom.opentext.lifecycle.action.execution property on TomEE:

1. Start <<TomEE installation folder>>\bin\TomEE.exe.
2. Click the Java tab.
3. Add the following text to the Java options:

-Dcom.opentext.lifecycle.action.execution=true

SOAP Request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <ExecuteLifecycleAction xmlns="http://schemas.cordys.com/entity/lifecycle/1.0">
      <ItemId>989096A082CBA1E89FA9C2D7BC15A4C7.16385</ItemId>
      <ActionName>SendToVerification</ActionName>
    </ExecuteLifecycleAction>
  </SOAP:Body>
</SOAP:Envelope>
```

Request parameters

Parameter	Description	Data Type
ItemId	Mandatory. Item Id of entity on which Lifecycle action must be performed.	String
ActionName	Mandatory. Name of the Lifecycle action.	String

SOAP response

When the operation is successful:

```
<ExecuteLifecycleActionResponse
  xmlns="http://schemas.cordys.com/entity/lifecycle/1.0">
  <result xmlns="http://schemas.cordys.com/entity/lifecycle/1.0" success="true">
    <info />
  </result>
</ExecuteLifecycleActionResponse>
```

When the operation is not accepted:

```

<ExecuteLifecycleActionResponse
  xmlns="http://schemas.cordys.com/entity/lifecycle/1.0">
  <result xmlns="http://schemas.cordys.com/entity/lifecycle/1.0" success="false">
    <info>The Event 'SendToVerification' could not be consumed by any of the
currently active states. </info>
  </result>
</ExecuteLifecycleActionResponse>

```

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 an application 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. Application 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.
Users can add this property to forms and lists, and use them as a part of the expressions for defining rules. Refer to this property in expressions using its fully qualified name. For example, the fully qualified name for the **TestDeadline** policy **Duedate** property is `item.Deadlines.TestDeadline_DueDate`.
- 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 a static duration or a duration property). If a duration is not specified, a due date is not calculated. The due date can also be configured for a specific item in the application.
- 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 email 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 instances.

Note: For a deadline Send Email action, you can select an email template that was defined on the entity. In the application, 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 building block:

1. In Workspace Documents, open the entity.
 2. Navigate to Available building blocks > Business logic and click **Deadline**.
The Add deadline dialog box opens.
 3. Enter the name of the deadline policy and click **Add**.
The Deadline designer opens in a new tab for configuration.
- Note:** You can view the added deadlines in the Added building blocks page. You can also edit the description, and open the deadline designer of the deadline policy using the details page.
4. 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 Browse icon appears. Click  (Browse) to open the expression editor and define the condition. See [Using expressions](#).
- 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 Browse icon appears. Click  (Browse) to open the expression editor and define the condition. See [Using expressions](#).
- c. In Default deadline, select **Static** or **Dynamic**.
 - If you select Static, specify the number of days, hours, and minutes.
 - If you select Dynamic, a list is displayed containing the duration type properties. Select the required property from this list.
5. Select the **Alerts and escalations** tab and click **Add action**.
 The Add Action dialog box opens.
6. In Add action, select **Send email**.
7. 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.
8. In Schedule, select **Static** or **Dynamic**.

- If you select Static, specify the number of days, hours, and minutes.
- If you select Dynamic, a list is displayed containing the duration type properties. Select the required property from this list.

9. Select **Before deadline** or **After deadline**.

For example, you might want an email to be sent two days before the deadline.

10. Click **Save** and close the window.

Important: For each deadline cycle, a process (BPM) called Entity Deadline Monitoring Process runs in the background in AppWorks 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 application users but a user with the Administrator role can view it from the Process Instance Manager. See *Instance Manager* in the *AppWorks Platform Advanced Development* 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.

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 email notifications are sent to assignees, and the email template to use for notifications. Each entity can contain only a single Assignee building block.

Important:

- Before you can configure email notifications, you must create the email 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 user's workgroups, setting the assignee to the user 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, an application 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 user individually or to one of the user's workgroups to another specific user or workgroup. This action displays a form prompting for the workgroup or user to be set as the assignee. This action requires Forward permission.

When an item is forwarded, assignee behavior works as follows in the application:

- 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 the 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. Navigate to Available building blocks > Business logic and then select **Assignee**. The Assignee building block is added to the Added building blocks pane.
3. Go to Details pane > Configuration.
4. From the When an item is created, assign list, select a value that is responsible for items.
5. Based on the value selected, select the remaining properties.

Note: You can specify using a property to set the assignment type when a user creates an item in the application. Accepted values for this property are Role, User DN, and Organizational unit.

Roles and users are set up in Open Text Directory Services (OTDS). In the application, 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.
- **Dynamic** enables selecting a property (such as user DN) whose value is used to set the assignee of an item in the application. 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
Organizational unit	Read from property	The assignee is read from an entity property, for example <<Property Name>>. The item property contains the name of the organizational unit.
Role	Static	The assignee is chosen from the Select a Role list.
	Dynamic	The assignee is read from an entity property, for example <<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 <<Property Name>>. The item property contains the user DN.
Read from property	Read from property	The type of assignee is read from the specified entity property, for example <<Property Name>>.
	Read from property	The assignee is read from the specified entity property, for example <<Property Name>>.

- 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.

6. Clear the **Record comments if the item is revoked by the assignee** check box if you do not want to record the comments.

Note: Comments are visible in runtime only if the Discussion building block is added to entity.

7. Click  (Save) and close the window.

Chapter 6

Designing the presentation

The presentation defines the user interface for the application.

Adding item layouts

Use the Layout building block to create the user interface for your application. A layout is a way to organize a set of panels into a page to present when a user opens an item from a list. 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. 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, 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.

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.

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 a form or a form named Default, users of your application get an error trying to view an item because the system cannot determine what form to use.

When you create an item layout, you can select Preview to display it in a Preview panel or Full to display it in a separate page. You can select both Preview Layout and Full Layout for an item layout. In many simple applications, an entity relies on the default item layout or provides a single layout that works well for all users.

When creating a list, you can select a Full Layout to use to display an item. If your application uses multiple home pages, you have the option to create multiple lists, each pointing to a different layout. For example, you might want to display different layouts for managers and data entry personnel. See [Adding lists](#).

To create an item layout:

1. In Workspace Documents, open the entity.
2. Navigate to Available building blocks > Presentation, and then click **Layout**.
The Add layout dialog box opens.
3. Type a name in **Name**, and click **Add**.
4. Populate the layout with the required panels. See [Layout panels](#) and [Adding panels to a layout](#).
5. 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 is 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.

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 Attachments](#)
- [Business Workspace](#)
- [Contents](#)
- [Create](#)
- [Discussions](#)
- [Emails](#)
- [External UI](#)
- [Form](#)
- [iHub](#)
- [Layout](#)
- [Lifecycle Progress](#)
- [Lists](#)
- [Preview](#)
- [Results](#)
- [Tasks](#)
- [Web Content](#)
- [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 [Adding 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.
%Complete	The percentage of activity flow completion, shown as a progress bar. The value of the progress bar depends on the number of completed activities.

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 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 space for breadcrumbs and to specify their location on the page.

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 so that the user can navigate back to the previous item viewed.

Business Attachments

The Business Attachments panel becomes available for configuration in a layout when you add a Business Workspace building block to an entity. Using the Business Attachments panel, application users can add existing content in Content Server to an entity. You can also use a Business Attachment panel in a child entity layout or related entity layout to manage workspaces created by the parent entity.

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. You can also use a Business Workspace panel in a child entity layout or related entity layout to manage workspaces created by the parent entity. 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. 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 Emails panel will be shown to users of your application. 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 AppWorks 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, AppWorks 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 items 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 AppWorks 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 of your application 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 [Filtering list results in the application](#).
- 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.
- Use the export option to export a list using the CSV or Microsoft Excel format.

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.

When the Results panel is initially loaded, no horizontal scroll bar is available. Columns are sized appropriately to fit in the panel without requiring a scrollbar. Users can display a horizontal scroll bar by clicking the Actions menu and selecting **Allow horizontal scroll bar**.

There are three ways users of your application can view information in the Results panel:

Normal mode without a horizontal scroll bar - As a user decreases the width of the Results panel, each column is also resized proportionally, but only to a predefined minimum width. As a user continues to decrease the width, some of the columns can no longer be displayed and are automatically hidden. As users increase the width of the Results panel, the hidden columns are displayed if there is enough room.

Normal mode with a horizontal scroll bar - The user has the option to have the horizontal scroll bar visible in the case where there are many columns and they are not all visible. When Allow horizontal scroll bar is enabled, the scroll bar is visible only when necessary to display all the columns.

Paragraph mode - Paragraph mode is used only when the width of the Results panel is too small to meaningfully display enough columns in Normal mode. The display is dynamically changed when the width of the Results panel is less than 480 pixels. The value of the data wraps to the next line if it is too long to be displayed. The display of a query or view form can be toggled on or off by clicking the funnel icon. When the form is visible, the paragraph grid is hidden to save space. When the view is executed or the form is closed, the paragraph grid is displayed and the form is hidden. There is no option for a horizontal scroll bar in Paragraph mode.

Users can personalize the Results panel in the following ways:

Specify the columns to display - Click the Actions menu, and select or clear columns, and click Apply.

Sort the items in a column - Click the column header. The sorting toggles between ascending and descending order. An up or down arrow icon is displayed on the title of the column that is used for sorting and the column name is shown in italics. Not all columns are sortable.

Note: The Long Text property cannot be sorted when using an Oracle database.

Specify the order in which the columns are displayed - Drag the column header left or right.

The next time the user opens the application, the panel remains in the state it was in when the previous session was closed.

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.

In the application, the following status icons are displayed in the Task panel.

Icons	Status
	Created
	Assigned

Icons	Status
	In Progress
	Complete
	Paused
	Suspended
	Obsolete
	Paused by system

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.

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 AppWorks Administration.
- 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: When the application runs, 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 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.
7. 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. Additional filter criteria can be added for the selected list. These filter the results based on the item context. For details on defining a filter, see [Defining a filter](#).

- **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.

8. 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, select the related entity and form to include.
- To include a Content panel, select the related entity.
- To include a Discussion panel, select the related entity.

You can include multiple panels from related entities in a layout.

Using panels in the application

Panels can be shown as tabs. Users can select a different panel by clicking the corresponding tab. Tabs that cannot fit in the window are available in a drop-down list.

If a user of your application chooses to print multiple panels, each panel is printed as a separate document. For example, if a layout has two forms and a discussion panel, the user has the option to print both forms and the discussions panel. If the user selects all three panels, a separate print preview is provided for each panel. Each panel is self contained and it is not possible to append each of them into a single print output.

Users can resize and maximize panels in the application. Users will only be able to maximize if you enable Chrome in the layout designer.

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 the application, the icon is automatically sized to fit the space. To maintain picture quality, the recommended size for the icon is 42px x 42px.

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 your application. For example, when a clinician selects a patient from a list, you can provide tabs 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 item layouts](#).

Adding forms

Typically, users view or enter information using a form. For example, a form can present the properties of the Order entity so that a user can create orders. Forms can organize information into various types of containers. They can also contain images and you can customize their appearance using formatting options. You can create as many forms as your business requires.

For each entity, you must create a form named Create. This form is used when a user creates a new item in the application. For example, if Create forms are created for the Vendor and Invoice entities, options called Invoice and Vendor are available from a menu when a 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 the application, 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 the application.

In addition to designing Create forms, You can also design other forms that are displayed when a 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, users get an error trying to view an item because the system cannot determine what form to use.

Users who have Create permission to create items must also have Update permission to enable them to update the properties in the item.

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, you need to remove and recreate the entire form.

Creating a form

To create a form, you first add the Form building block to the entity. Then you add properties and other components to it and customize them based on your requirements. 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 (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/start/web/
perform/create/00505601050711E6FAD6E59DA5E516C1?rurl=
home/system/app/start/web/perform
```

The following components are available for configuring a form.

Component	Description
Property	Defines the information that users 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 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.

Component	Description
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. Navigate to Available building blocks > Presentation, and then click **Form**.
The Add form dialog box opens.
3. Type the form name in Name and click **Add**.
The form designer opens.
4. 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](#)
5. 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 the application. See [Including multiple forms in a layout](#).

Printing forms in the application

Users of your application can print forms using the action bar, the Print icon in the global header, or action buttons that you place on the form. They cannot print from a List action bar or a Home Page Layout. They can also specify options for what will be printed. For example, a user can print a form that provides information about a specific order.

When using the Print option from the action bar or the global header, users have the following options:

- Include the header logo within the printed form.
- Print all icons for radio button options or just the selected option. If they choose to print all options, all option groups (both those that are selected and those that are cleared) are printed. If they choose not to print all options, only the selected option is printed. If they choose to print all options, every option (and its selection icon) is printed.

Printing to PDF format is available by default using Chrome. Firefox can use a free addon such as PDFCreator.

The option to print in landscape orientation is available in Chrome. The landscape option is not available for Firefox and Internet Explorer.

Note: Your application can print only what the user sees on the screen.

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 an application 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. 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, background color, and style. You can also select the text alignment for a text box presentation.
 - Select **Required** if a user must enter a value for the property. Select **Read Only** if a user cannot change the value for the property.
 - Select or clear **Show Label** to specify whether to show a label for the property in the application. 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, background 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 [Using toolbars and icons](#) 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.

- Customize the numeric property types such as decimal, float, and integer, using the multiple formatting options. When the default formatting option is selected, the presentation format in your application is derived from the locale of the user. To customize the format, clear the option and set the following options for the presentation:
 - **Negative values:** Value that 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. This option is not available for the **Integer property type**.

Note: The option to change the **Background color** is available only for the Text box, Text only, and Multi-line presentation types.

- 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. Also 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.
 - For Date and Date and time property types, see [Defining the format for Date and Date and time properties](#) for a description of the formatting options.
4. If you want to associate a property with a category that will be disabled or hidden in the application, select it and then select or type a name in **Category**.
See [Disabling or hiding information on a form](#).
 5. Click  (Save).

Defining the format for Date and Date and time properties

When you add a Date or Date and time property to a form, you have various options for specifying how it is displayed to users. You can use a predefined format or you can create a custom format. The default presentation format is short format (the date and time presentation is separated by a space delimiter) and the date and time is displayed based on the user's locale.

Predefined formats

The following predefined formats are provided. In your application, all predefined formats except ISO 8601 are displayed based on the user's locale. The EN_US and nl_NL locales are used as examples.

Predefined format	Examples - Date and time	Examples - Date
Short	8/16/2023, 3:45 PM (en_US) 16-8-2023 15:45 (nl_NL)	8/16/2023 (en_US) 16-8-2023 (nl_NL)
Medium	Aug 16, 2023, 3:45:00 PM (en_US) 16 aug. 2023 15:45:00 (nl_NL)	Aug 16, 2023 (en_US) 16 aug. 2023 (nl_NL)
Long	August 16, 2023 at 3:45:00 PM (en_US) 16 augustus 2023 om 15:45:00 (nl_NL)	August 16, 2023 (en_US) 16 augustus 2023 (nl_NL)
Full	Wednesday, August 16, 2023 at 3:45:01 PM (en_US) Woensdag 16 augustus 2023 15:45:01 (nl_NL)	
ISO 8601	2023-08-16 03:45:00	2023-08-16

Custom format

To create a custom format, provide the required elements in the given sequence from top to bottom in the following table. The system automatically displays the localized values based on the locale of the user in the run time.

You can define a custom format by selecting **Custom (localized)** and specifying the format as described in the following table. The default value for the Custom (localized) format is yyyyMMMdEEEdjmss.

Custom format	Examples
yyyyQQQ	Q3 2023 (en_US) K3 2023 (nl_NL)
MMMdEd	Wed, Aug 16 (en_US)

Custom format	Examples
	wo 16 aug, (nl_NL)
MMMEddHHmm	Wed, Aug 16, 15:45 (en_US) wo 16 aug. 15:45 (nl_NL)
ymmss	3:45:01 PM (en_US) 15:45:01 (nl_NL)

The following table describes the components of the date format.

Field type	Symbol	Field pattern	Example
Year	y	yy	23
		yyyy	2023
Quarter	Q	Q	3
		QQQ	Q3
		QQQQ	3rd quarter
Month	M	M	8
		MM	08
		MMM	Aug
		MMMM	August
Weekday	E	EEE	Tue
		EEEE	Tuesday
Day	d	d	1, 16
		dd	01, 16
Hour	h	h	1 a.m., 12 p.m. (nl_NL) 1 AM, 12 PM (en_US)
		hh	01 a.m., 12 p.m. (nl_NL) 01 AM, 12 PM (en_US)

Adding relationships to a form

By adding a relationship to a form, you can include properties from the related entity on the form and specify the available actions that are available for those properties in the application. 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 the icon 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 the application, it is best practice to create separate "internal" lists that include only the properties that are required for 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 the application.

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** - 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. You cannot change the presentation of individual properties inside the container. To select an item, a user makes a selection from each drop list in succession.

In the following example, after the user makes a selection from Make, Model becomes enabled. After selecting a Model, Year becomes enabled.

A screenshot of a form interface. At the top left is a text input field labeled 'CustomerName' with the value 'John'. Below it is a section labeled 'Entity Picker Container' containing three dropdown menus: 'Make' (set to 'Toyota'), 'Model' (set to 'Camry'), and 'Year' (set to '2009'). The 'Year' dropdown has options '- Select -', '2009' (selected and 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 the application.

Note: A property used inside a drop list series must support sorting and comparison operations. The property should not be of a large object type, such as Image, Long Text, or any binary or text large object type if the entity is an imported entity from a database.

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 the application. 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 the application, 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 the application.
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 the application.
 - Browse - The user can find and select an item from the related entity. For a To one relationship, this option is selected by default and cannot be cleared.
 - Create - The user can create an item.
 - Clear - Data is cleared, but it 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 the application, select it and then type a name in the Category box in the right pane.
See [Disabling or hiding information on a form](#).
Children cannot be browsed or cleared, so Browse and Clear options 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).

You can alternatively drag and position a related property and the relationship to the form individually and then configure each individually.

Important: In the application, a related property on a form is disabled until the relationship is established by clicking  (Browse). 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  (Browse). 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, users of your application 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.

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 application users in the following ways. After you save the form, you cannot change the presentation. For any repeating group container nested inside another repeating group container, the presentation is the 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 an application. 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 also 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
16,385	10/02/2008 12:00 am
Number	Employed
25	<input checked="" type="radio"/> True <input checked="" type="radio"/> False <input type="radio"/> Unknown
Name	Frank

▼ 2

Id	Started Time
16,390	05/12/2015 12:00 am
Number	Employed
25,221	<input checked="" type="radio"/> True <input checked="" type="radio"/> False <input type="radio"/> Unknown
Name	Chris

▶ 3

▶ 4

The information for each item 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, users can access 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 the application, the title row shows a row number whenever possible. For a parent child or a bidirectional One to many relationship, the row number is assigned based on the order of the related entity primary key. When the related entity is a native entity, the primary key is in the same order in which the item was created (the oldest item has row number 1).

When a new item is added to the container using either the Create or Browse action, the newly added item is added to the top of the container so that users can easily work within the current view. The title row of a new item is shown in green to indicate that it is temporarily placed out of order. Once the form is reloaded, items are then displayed sequentially in the order of the primary key.

The row number is not guaranteed in any specific order for a unidirectional Many to many relationship.

The title row also shows additional information based on the entity property value, in this order:

- If there is a Title building block in the related 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 and 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.

The title is shown as <Title> and the first required property (preferably text) or row number is shown as a tooltip.

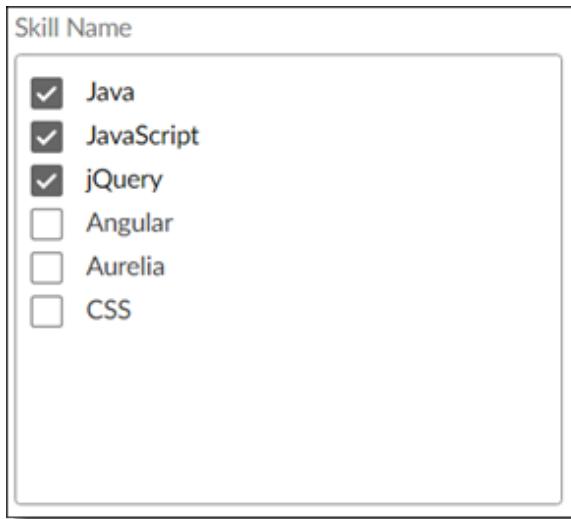
Grid - Presents the related entities in a spreadsheet format, with each related entity in a row and its properties in columns. Using the Grid presentation, users can add a new row, delete one or more rows, show or hide columns, sort data on columns, and filter data.

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 the application, users can select Browse, Create, and Clear actions from the drop-down menu of the grid cell. After the related item is selected, the property value of the selected item is displayed in the grid cell.

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 an application. Users can link or unlink one or more items by selecting or clearing the check box, but they cannot create new items.



The Multiselect presentation is not shown on mobile devices.

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**. 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 the application with all rows expanded. The default selection is **Collapsed**.
 4. If you want to display the number of items in the container, select **Display number of items**.
- Caution:** The item count may become inaccurate in the application in the following situations:
- For a parent-child relationship, the total item count shows an accurate count, unless conditional security is defined that makes certain child items not visible to the current user.
 - For a peer-to-peer relationship, with the delete behavior defined as not allowing the users to delete a related item if referenced by another item, the total item count is accurate, unless conditional security is defined that makes some items not visible to the current user.
 - For a peer-to-peer relationship, with delete behavior defined as allowing the user to delete a related item if referenced by another item, the total count becomes

inaccurate as soon as the items are deleted outside the container. In this case, it is recommended not to use this option in your form.

5. If you want the repeating group container to automatically resize 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](#).
6. In **Actions**, select the options to be available on the form in the application. 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 related 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.

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.

7. 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 an application 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.
8. If you want to associate the container with a category that will be disabled or hidden in the application, 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 the application. 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. Content formatted with the Rich text presentation will have HTML tags when shown in other presentation types and in List columns. 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**.
 - Image – Enables users to upload and view images 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 user must enter a value for the property. Select **Read Only** if a user cannot change the value for the property.
 - Select or clear **Show Label** to specify whether to show a label for the property in the application. 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 the application, 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.

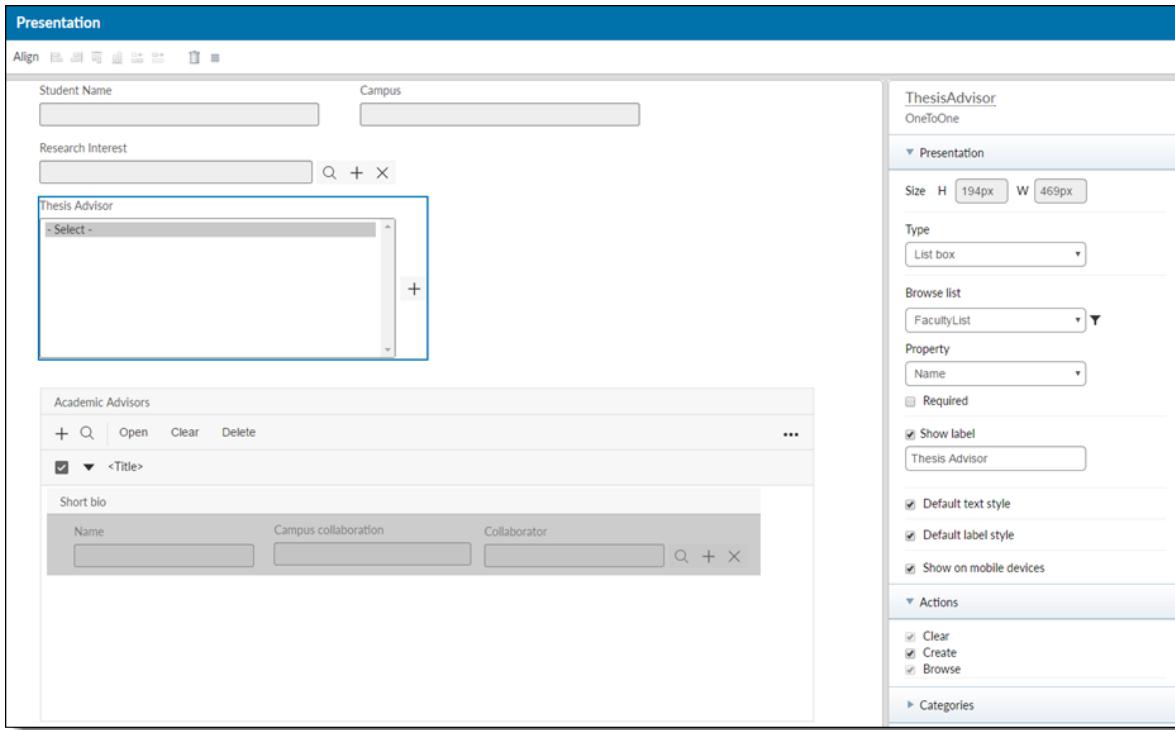
When designing a form, you can associate a relationship control with a filter that contains context variables.

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 not filled 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 ▼ (Filter) in the **Browse list** area, you can specify the filter to use.



To define a filter, see [Defining a filter](#).

Defining a filter

An application often displays limited items based on the filter criteria. Use the filter beside the selected list to configure the filtering criteria.

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.

The property must be searchable in the browse list. A list property used in the filter must be defined as **Include as filter**.

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 the application, 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
eq	Property values are equal to the parameter value	This is the default operator if an operator is not specified,
ne	Property values are not equal to the parameter value	
lt	Property values are less than the parameter value	
gt	Property values are greater than the parameter value	
ge	Property values are greater than or equal to the parameter value	
le	Property values are less than or equal to the parameter value	
like	Property values are "like" the parameter value, using the database server's wild card capabilities	<code>item.Properties.StudentName=Like(A%)</code>

Operator	Description	Notes
InList	Property values are among a specified list of possible values	item.Properties.Campuse=InList(Berkeley,LA,SF) no spaces before or after comma.
NotInList	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	Property values are between the specified bounds	item.Properties.Value=Between(13,99) no spaces before or after comma.
Null	Property values are null	Properties.Published=Null()
NotNull	Property values not null	Properties.Published=NotNull()

Note: You can define advanced filters using the following wildcards in combination with the **like** comparison operator:

Wildcard character	Description
*	Use the asterisk to represent zero or more characters. For example, to find values that start with Alpha , type Alpha* .
?	Use the question mark to represent a single character. For example, to find values that have U as the second character, type ?U* .

Working with filters defined on a list

The filters defined for a selected list work together with the filters defined within 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 individually 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 the application. 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 the application. 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 tabs 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. Click **Components > Other > Tab Container**.
3. Drag the container to the location where you need it.
4. In **Number of tabs**, type a value.
5. 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.
6. 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**. You can use categories in rules to disable or hide information in the application. See [Disabling or hiding information on a form](#).
 7. Drag properties to each tab and configure their attributes. See [Adding properties to a form](#) and [Adding relationships to a form](#).
 8. 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/start/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 the application.
 - 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 user of the application clicks an action button, the action is invoked. If the action requires parameters, the user is prompted to enter them and they are passed 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 the application.

You can customize styles for an action at the form level. These styles are applied to the action buttons within the form.

Note: The applied styles override any default AppWorks Platform styles or customized styles being used. This ensures consistency in the styles being used.

For imported entities, actions are not shown in a form when it is launched from the Create action in the application.

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. For a Create form, only Rule, Upload, Download, and Title actions are supported.

3. Update the following properties if required:
 - The label that identifies the action. The label is editable and supports translation.
 - The tooltip that is displayed to application users. By default, this is the same as the label. The tooltip is editable and supports translation.
 - The 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 specify 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).

To style an action:

1. Select the form.
2. In the **Form > Presentation** pane, clear the **Default button style** check box.
3. Update the following attributes:
 - **Border color**: Color of the borders of the action button.
 - **Border radius**: Radius of the button corners.
 - **Button color**: Color of the action.
 - **Label font**: Font style of the action label.
 - **Size**: Font size of the action label.
 - **Color**: Font color of the action label.
 - **Style**: Text style of the action. Options are Bold, Italic, and Underline.
4. Click  (Save).

Adding images and color to a form

By including images and color in your organization's forms, you can give the forms a distinctive look and feel that is customized to your specific 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 the application, you need to 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. 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 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**.
 - In **URL**, type the URL for the link.
 - 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 the application. 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 the application. 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. This rendering method is required for certain types of applications, for instance when a form is overlaid on an image. For other applications you will typically want large and complex dynamic 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 a form is a horizontal region that extends the entire width of the form canvas. You add a new section to divide the 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 previously 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  (Insert section before) or  (Insert section after).

To move a section:

- Select the section and then click  (Move section up) or  (Move section down).

To remove a section:

- Use the same procedure that you use to remove containers and other 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 displays a responsive form interface. At the top, there are 'Print' and 'Delete' buttons. Below them is a row of fields: 'Id' (32770), 'Urgency' (NotSpecified), 'Assignee' ('Select'), and two radio buttons for 'Show Related' (Yes is selected). A large text area labeled 'Description' is followed by a radio button for 'Show Watchers' (Yes is selected). The main content area is divided into sections: 'RelatedTickets' and 'Watchers'. The 'RelatedTickets' section contains a header row with 'Id' and 'Description' columns, a data row for ticket 32769, and a summary row with a count of 2. The 'Watchers' section follows a similar structure. At the bottom is a 'Comments' section with a text input field. The entire form is designed to be responsive, with the repeating group containers (RelatedTickets and Watchers) adapting to the available space without scroll bars.

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 property.

To prevent a user from replacing a related property with another value, delete the Search icon from the form. To prevent a user from changing a property based on a specific 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.

Event

When A property changes ▾

i This rule will run on loading a form or when a property used in the following condition changes.

Condition

Basic Advanced

If all of the following are true

Item equal to Laptop

Action

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 the application.
 - 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. By creating a library of mini-forms that you intend to nest in other forms, you can save a considerable amount of time and effort when creating forms. 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.

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 containers 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 the application 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 the application.

The screenshot shows a form design interface with three columns:

- Credentials:** Contains fields for "User Name" and "Password".
- Full Name:** Contains fields for "First Name" and "Last Name".
- Address:** Contains fields for "Street Address", "Apt/Unit", "Zip", and "State".

At the bottom of the form are two buttons: "Create" and "Cancel".

Positioning components on a form

You have the following options for positioning components on a form.

Option	Procedure	Comments
Move a component	Drag it to another location on the Presentation panel or use the Keyboard shortcuts for forms .	The height of the following components is automatically resized based on the font and you can resize only their width: Text box, Text only, Multi-line, Date, Date Time, Check box, Drop down list, List box, and Radio button. You also cannot resize the height of the Horizontal Line.
Resize a component	Drag its sizing handle or borders or use the Keyboard shortcuts for forms .	The selected component is resized.
Resize multiple selected 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.
Align selected 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 enable creating forms with multiple columns, the alignment operations intentionally align components relative to the first component selected rather than to the workspace.
Delete selected	Click	The components are removed from the

Option	Procedure	Comments
components	selected components) on the toolbar.	form.
Toggle the display of the alignment grid	Click  (Toggle Grid) on the toolbar.	The alignment grid is shown or hidden.

Keyboard shortcuts for forms

When designing forms, the following keyboard shortcuts are available for commonly used functions.

Shortcut Key	Function
$\leftarrow, \rightarrow, \uparrow, \downarrow$	Move the selected components to the next five pixel boundary in the specified direction.
CTRL+ \leftarrow , CTRL+ \rightarrow , CTRL+ \uparrow , CTRL+ \downarrow	Move the selected components one pixel in the specified direction.
SHIFT+ \leftarrow , SHIFT+ \rightarrow	Increase or decrease the width of the selected components by 5 pixels by moving the right edge of the component.
SHIFT+ \uparrow , SHIFT+ \downarrow	Increase or decrease the height of the selected components by 5 pixels by moving the bottom edge of the component.
ALT+ \leftarrow , ALT+ \rightarrow , ALT+ \uparrow , ALT+ \downarrow	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

Shortcut Key	Function
	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 of 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 label 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 progress 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.

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. An application 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:

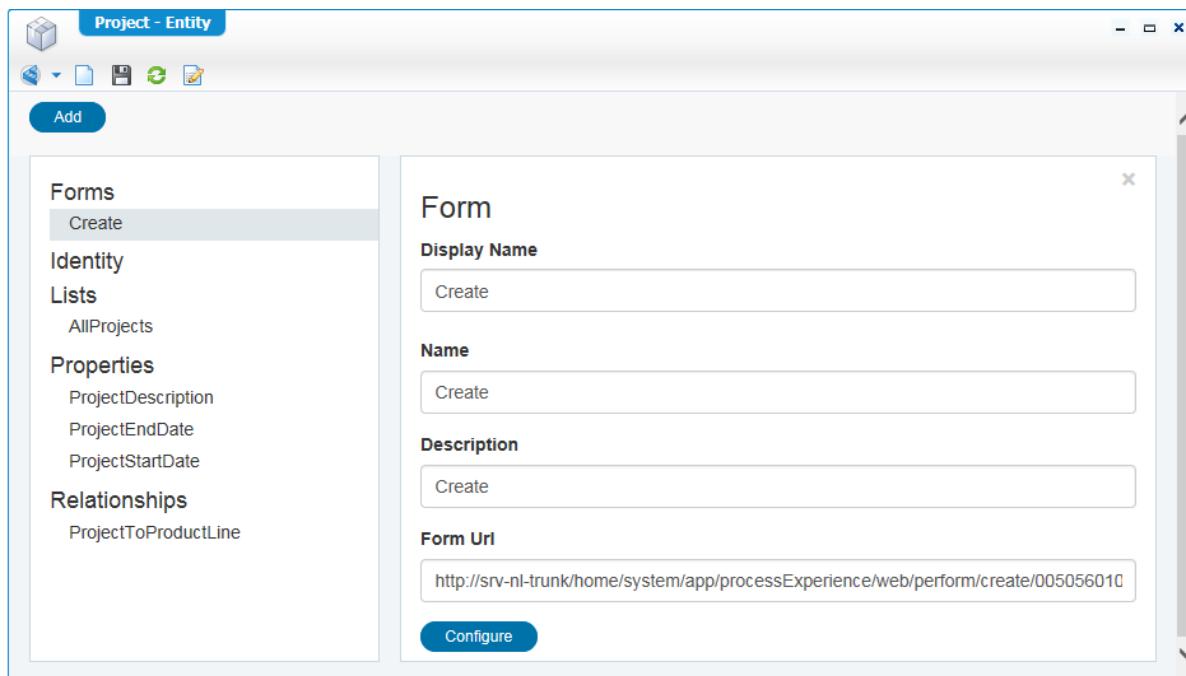
- Project
- Product
- ProductLine

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.

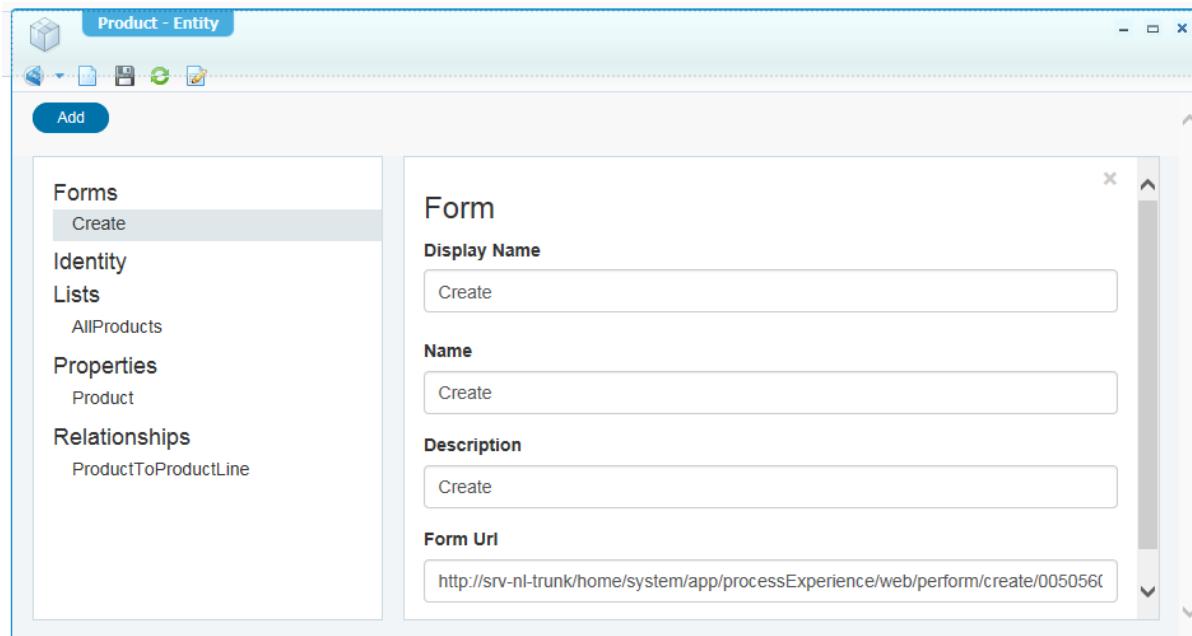
Entity configuration

Each of the entities must have properties, forms, lists, and relationships, as shown in the following screenshots.

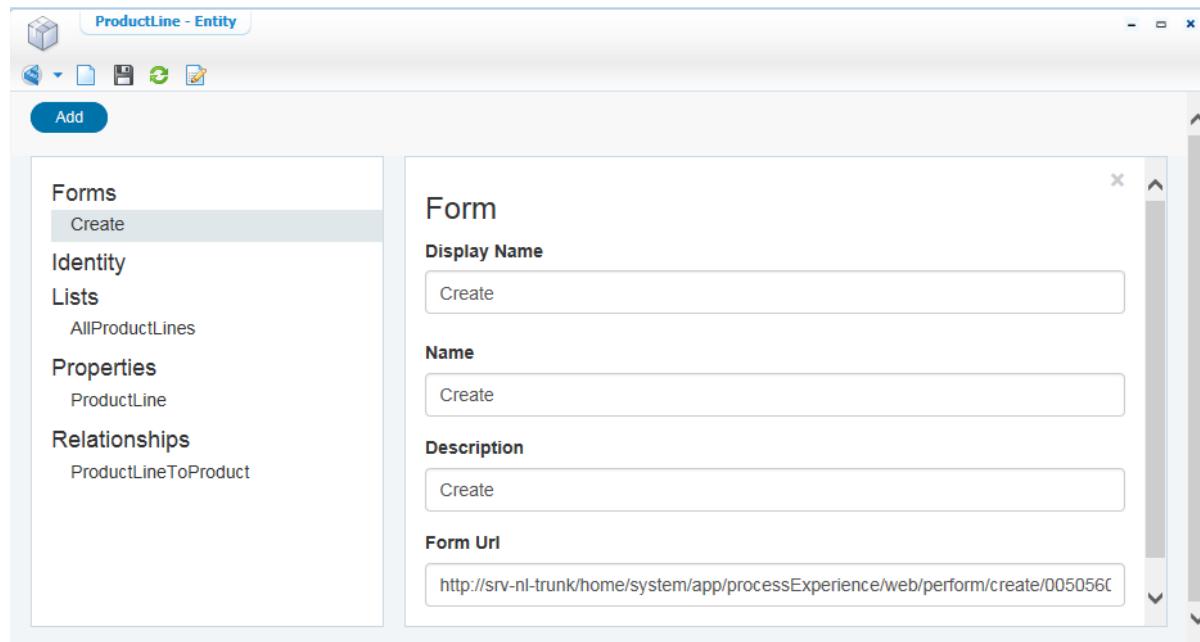
Project entity



Product entity



ProductLine entity



Building block configuration

The following steps are required to get the desired results with a drop list series. 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 bidirectional. (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 the Project entity, edit the default Create form or create a new one as follows.
 - a. Drag the To one relationship called ProjectToProductLine to the form and change the presentation to Drop List Series.
 - b. Select the Browse List (in this case, the list is called All Product Lines).
 - c. Select the Product property and click **Add** to add the Product Line property.

ProjectToProductLine

OneToOne

▼ Presentation

Size H 130px W 656px

Type

Drop list series

Browse list

All Product Lines

Property 1

Product

Property 2

Product Line

Add

- d. In the Product and ProductLine entities, add Create forms. In the application, 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.

In addition to building a Create form, you can also build a Default form for each entity to view items.

- 5. Publish the project and, if necessary, assign access rights in [AppWorks Administration](#).

Application flow

In the application, 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.

The screenshot shows a 'Create Project' dialog box. At the top, there are three input fields: 'Project Description' (containing 'Inventory'), 'Project Start Date' (set to '02/20/2017'), and 'Project End Date' (set to '02/22/2017'). Below these, there is a section titled 'ProjectToProductLine' containing two dropdown menus. The first dropdown under 'Product' is set to 'Computer'. The second dropdown under 'Product Line' has a blue header bar with the text '- Select -'. Below this header, four options are listed: 'Desktop', 'Laptop', 'Notebook', and 'Tablet'.

Adding lists

Use the List building block to configure lists that will enable users to display and find items in the application. Lists are the most common way for users to interact with the system. Users access a list to display items and then open or work with the items.

In your application, users can filter lists to narrow the results. See [Filtering list results in the application](#).

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 using entity modeling). 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 [Accessing information in external systems](#).

To add a list to an entity:

1. In Workspace Documents, open the entity.
2. Navigate to Available building blocks > Presentation, and then click **List**.
3. In the Add list dialog box, type the **Label** and **Name**, and then click **Add**. The list designer opens.
4. On the Properties tab, select the required properties.
5. Click the Filters tab and configure filters. For details, see [Designing a list](#).
6. To provide more details, go to the list details pane.

- In Configuration:
 - Edit the **Label**.
 - In the Action bar, select an action bar to show for the list. In the application, any actions that are not relevant are not available. See [Creating an action bar](#).
 - In Layout to use when opening an item, 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.
 - If you want to open the designer for an existing list, click **List designer**.
 - In Advanced configuration:
 - In the Default category, specify an (optional) default category for the list. 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.
 - In Number of results per page, select All or specify the maximum number of items to show in the list by selecting the Limited to option. If the list returns more than this number of results, the user can page through them.
 - If you want the list to be visible to application users, select **Show this list to end users**.
 - If you want users to be able to create items from the list, select **Allow users to create items**. In your application, the Create action (+) button is visible on the results list only if this option is selected and the required permissions are granted using the security modeler. This option is available only when Show this list to end users is selected. This option is not available for a list defined on a child entity.
 - Select **Show total count in results** to display the count of available items in the Results panel.
3. Click  (Save).

Designing 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 user of the application 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	A;B;C	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.

Note: All the to-many relationship properties are also displayed. However, further relations under these to-many relations are not displayed. These properties are available for filtering, but cannot be viewed in the Results panel.

3. 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.

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: Label, 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 minimum and maximum values in the property configuration. If the minimum and maximum values are not specified, by default 0 and 100 are considered as the minimum and maximum values. For example, assume that you configure a property with a Minimum value of 0 and a Maximum value of 50. If a user of the application 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 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.
- Note:** The Long Text property cannot be sorted when using an Oracle database.
- 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.
- Note:** For any to-many relationship property, this option is selected and disabled.
- For a Date or Date and time property type, see [Defining the format for Date and Date and time properties](#) for a description of the formatting options.

On the Filters tab, configure filter rules to apply 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 [Advanced functions](#) for a list of advanced functions that you can use to specify a filter for a list.

2. 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	Currency	Date	Date Time	Decimal	Float	Integer	Long Text	Text	Enum Text	Enum Integer
Equal	X	X	X		X	X	X	X	X	X	X
Not Equal	X	X	X	X	X	X	X	X	X	X	X
Greater Than		X	X	X	X	X	X				
Greater Than or Equal		X	X	X	X	X	X				
Less Than		X	X	X	X	X	X				
Less Than or Equal		X	X	X	X	X	X				
Between	X	X	X	X	X	X	X	X	X		
Within			X	X							
Contains								X	X		
Any Of		X			X	X	X	X	X		
Empty	X	X	X	X	X	X	X	X	X	X	X
Not Empty	X	X	X	X	X	X	X	X	X	X	X

Note: You can define advanced filters using the following wildcards in combination with the **contains** filter operator:

Wildcard character	Description
*	Use the asterisk to represent zero or more characters. For example, to find values that start with Alpha , type Alpha* .
?	Use the question mark to represent a single character. For example, to find values that have U as the second character, type ?U* .

Performance considerations

This section describes some ways you may be able to improve performance for lists.

Creating indexes on columns

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 degrade performance. This is also relevant for Find web service operations.

Entity instance security tables

When using the Sharing building block in any of your entities, for performance reasons, create these indexes for the columns in the following tables:

Table name	Column name
O<YourOrganizationId>OpenTextEntityInstanceSecurityAclEntry	RoleId
O<YourOrganizationId>OpenTextEntityInstanceSecurityMergedAcl	CascadeDenials, CascadeGrants
SHRDOpenTextEntityInstanceSecurityAclEntry	RoleId
SHRDOpenTextEntityInstanceSecurityMergedAcl	CascadeDenials, CascadeGrants

Note: These tables are created on creating an organization.

Maximizing performance for SQL Server

Execution of a list with a Long text property using the Contains operator (a filter you set in the list configuration or a filter set by the end user in the application) requires creation of a full text index to maximize performance in SQL Server. This performance optimization is available only for SQL Server and enabled via

feature.toggle.entitymssql.db.fulltext.search, which is disabled by default. The entity publish/deploy logic does not automatically create the full text index for a Long text property. Follow the steps in the SQL Server article [<https://docs.microsoft.com/en-us/sql/relational-databases/search/use-the-full-text-indexing-wizard?view=sql-server-2017>] for defining a full text index.

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

You can use these functions 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

Working with tasks in application lists

A task is a message sent from an action to application users indicating that a certain activity has occurred or providing an instruction to perform the next step in a process. Users can either respond to the task, add a new action to the flow after the task, or do both. For example, when the stock of a particular item in a warehouse goes below a certain level, a re-order task is sent to the warehouse manager. On receiving the task, the warehouse manager opens it, fills in a purchase order form, and forwards it to the purchasing manager.

Users can work on the tasks available in the personal list and they can claim tasks that are assigned to the roles, teams, or worklists that they are associated with. If applicable, users can view the tasks, open them, and complete them directly from the Inbox. The following lists are available to users:

- All Tasks
- Personal Tasks
- Roles Tasks
- Teams Tasks
- Worklist Tasks

User operations

From the Inbox, users can perform various operations on tasks. For example, they can claim tasks, initiate claimed or assigned tasks, pause tasks, and so on. All users can perform some basic operations. A Work List Manager or a Team Lead can perform additional operations, such as, skipping a task, suspending a task, or forwarding a task to another work list or team and so on.

The following table describes the various operations that users can perform.

Action	Description	Status
Claim	Claims the task from the work list. Any user in the list can claim the task and work on it. Once the task is claimed, it is displayed under the Personal Tasks tab in the Inbox of the user and the corresponding list.	The task status changes from New to Assigned.
Start	Starts executing the claimed task. Once the task is started, the user can Stop, Pause, or Complete it.	The task status changes from Start to In-Progress.
Pause	Temporarily stops executing the task. A Resume option becomes	The task status changes to Paused.

Action	Description	Status
	available to resume a paused task.	
Stop	Completely stops executing the task. Click to start executing a stopped task.	The task status changes to Assigned where the user can start the task again.
Revoke	Revokes a claimed task. Once revoked, the task is moved back to the corresponding folder. For example, a revoked task is moved back to the unassigned state. Note: Only claimed tasks can be revoked.	The task status changes to New.
Complete	Completes executing the task. See Complete .	The task status changes to Complete. In addition, the user's tasks are represented with the following icons if they were configured in your application to include them: <ul style="list-style-type: none"> ■ Recommended – Thumbsup ■ Not Recommended - Thumbsdown ■ Warning -Warning icons
Skip	Skips the task. Once the task is skipped, it becomes a non-workable task. See Skip .	The task changes to Obsolete.
Delegate	Delegates the task to another user. See Delegate .	This is performed on the personal tasks. The ownership of the task lies with the User who delegated the task. For the procedure on delegating a task, see Delegating a Task .
Forward	Forwards the task to another user. See Forward .	This is performed on the personal tasks. The ownership of the task lies with the User to whom the task is forwarded. For the procedure on forwarding a task, see Forwarding a Task .
Modify start date	Modifies the start date. Notes: The Start Date should be earlier than the Due Date of the selected tasks and greater than the	

Action	Description	Status
	current date. The Start Date of an In-Progress task or activity cannot be modified.	
Modify due date	Modifies the due date. Note: Due Date should be later than the Start Date of the selected task.	
Add tasks	Adds the tasks in the current state or plans the tasks for the future state on an ad-hoc basis, when required. See Add tasks .	These tasks are represented with the following icons if they were configured in your application to include them: <ul style="list-style-type: none"> ■ Recommended – Thumbsup ■ Not Recommended - Thumbsdown ■ Warning -Warning icons
Add related task	Adds related tasks for a selected task on an ad-hoc basis, when required. For more details, see Add Related Task .	These tasks are represented with the following icons if they were configured in your application to include them: <ul style="list-style-type: none"> ■ Recommended – Thumbsup ■ Not Recommended - Thumbsdown ■ Warning -Warning icons
Properties	Displays the properties of a task.	

Manager operations

In addition to the above operations, a Work List manager or a Team Lead can perform the following operations.

Action	Description	Status
Assign	Assigns the task to a user. See Assign .	The task status changes from New to Assigned.
Complete	Completes executing the task on behalf of any user, if required. See Complete .	The task status changes to Complete.
Suspend	Suspends the 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. Once suspended, a user	The task status changes to Suspend.

Action	Description	Status
	can click Resume to resume the task.	
Skip	Skips the task. Once the task is skipped, it becomes a non-workable task.	The task status changes to Obsolete.
Forward	Forwards the task to a user. This is performed on the tasks in the list or team folder. The ownership of the task lies with the User to whom the task is forwarded. For the procedure on forwarding a task, see Forward .	This is performed on the personal tasks. The ownership of the task lies with the User to whom the task is forwarded. For the procedure on forwarding a task, see Forwarding a Task .

Assign

Assigns the task to a user. Users with a work list manager role or a team lead role can assign a task to other users in their corresponding work lists or teams. Once a task is assigned to a user, the task becomes a personal task for the user. The Assign Task dialog box contains the following information.

Field	Description
Name	Name of the task on which this action is being performed.
Assignment type	The assignment target type. Select from one of the options: <ul style="list-style-type: none"> ■ Role ■ Worklist ■ Team
Assigned to	Target name to which the current task is assigned. Only users of the current assigned target are displayed in the list.

Add tasks

Adds the tasks in the current state or plans the tasks for a future state. This option shows the list of tasks in various states that can be added on an ad-hoc basis when required. The Add tasks dialog box displays the following sections:

- **Choose a state to add tasks** - List of states for the item. On selection, the available ad-hoc tasks for that item state are shown. This lists only the states that have ad-hoc tasks. By default, the current state of the item is selected.

Note: Users can add tasks for all states. However only tasks in the current state of the item are available to add immediately. The remaining added tasks will be available only when an item progresses to the respective state.

- **Available tasks** - The available tasks of the selected state. Users can select a task and configure further information provided in the Configuration section.
- **Configuration section** - Enables users to configure the attributes such as Assignee Type, Assignee value, and Duration for a selected task.

The following parameters are available in the configuration section.

Field	Description
Name	Name of the task on which the selected action is being performed. This can be modified before adding the task.
Assignment Type	Assign the task to one of the following types: <ul style="list-style-type: none"> ■ Assign to me ■ Individual ■ Role ■ Team ■ Worklist
Assigned to	The values for the assignment depends on the assignment type selected. Based on the assignee type selected, one of the following values are populated in the drop-down list: <ul style="list-style-type: none"> ■ Work list - List of worklists ■ Team - List of teams ■ Role - List of roles ■ Individual - List of users Select the corresponding value from the drop-down list.
Assignee	Users for the selected worklist, team, or role. This is not applicable for Individual assignment type.
Due on	Configures the time by which the task should be completed. Users can provide the duration or a fixed date to configure the time. <ul style="list-style-type: none"> ■ Duration - Displays options to set Days, Hrs, Mins. ■ Date - Displays the calendar to choose the date from.

Select the required tasks with configured information and click **Add** to initiate the selected tasks.

Notes:

- All tasks are populated with default configurations (if any), such as assignments and due date. By default, the current user is assigned tasks that have no pre-assignments.

- While adding tasks, users can overwrite the default configurations. For each task, users can overwrite any configuration changes using the **Set default values** option.
- If the assignment information is not configured for a task, it is assigned to the user who added the task.
- The Available tasks section also lists the processes that may be a system defined service or a set of predefined tasks. No configuration is required for these processes.

Add Related Task

Users can add related tasks for the selected task. When these tasks are added, the current task is paused and awaits the completion of the related tasks. After the related tasks are completed, the current task becomes active and the user can resume working on it.

Related tasks can be used to get further information for a task or work to be done for performing the selected task. Availability of related tasks depends on your application design.

This following options are displayed in the dialog box:

- **Available Related Tasks for Current Task:** Shows all the available related tasks for the current task. On selecting a task, the corresponding configuration section is displayed on the right-side.
- **Configuration for selected task:** Enables users to select attributes such as Assignee Type, Assignee Value, Duration, and so on. See the configuration section in [Add tasks](#) for details.

Forward

Tasks can be forwarded from one user to another or across teams, work lists, and roles. While users can forward tasks to other users within their group, work list managers or team leads can forward tasks across work lists or teams.

Note: While forwarding a task, the ownership of the task is transferred to the receiver.

Task can be delivered to different types of targets such as lists, roles, teams, and users. When a task is delivered to a specific target type, the Forward Task dialog box shows various options based on the authorizations to the users and target type to which the task is delivered.

Assignment Type	Authorization	Action options
List	Work List Manager	<p>A Work List Manager can also forward tasks across other worklists or to users in the assigned work list.</p> <p>The available options are based on whether the task is assigned or unassigned.</p> <p>To forward a task to a work list or user:</p>

Assignment Type	Authorization	Action options
		<ol style="list-style-type: none"> 1. Select the Worklist option and then the respective worklist, or select the User option and then select the user. 2. In Reason for Forwarding, provide the reason for forwarding the task. When forwarding a task, a reason must be provided.
Team	Team Lead	<p>A Team Lead can forward tasks across other teams in the organization or to users in the assigned team. The available options are based on whether the task is assigned or unassigned.</p> <p>To forward a task to a team or user:</p> <ol style="list-style-type: none"> 1. Select the Team option and then the team, or select the User option and then the user. 2. In Reason for Forwarding, provide the reason for forwarding the task. When forwarding a task, a reason must be provided.
Role	Role	<p>A user with the required role, can forward tasks to other roles in the organization or to users of the assigned role. The available options are based on whether the task is assigned or unassigned.</p> <p>To forward a task to a role or user:</p> <ol style="list-style-type: none"> 1. Select the Role option and then the role, or click the User option and then the user. 2. In Reason for Forwarding, provide the reason for forwarding the task. When forwarding a task, a reason must be provided.
User	User	Select the user from the drop-down, provide the reason, and then click OK .

The following information is displayed in the dialog box.

Name	Description
Task name	Name of the task on which the action is being performed.
Assignment type	The assignment target type: Role, List, or Team.
Assigned to	The name to which the current task is assigned.
Forward to	List of users to be forwarded.
Reason for Forwarding	The reason for forwarding the task.

Delegate

Tasks can be delegated to other users for several reasons. For example, a user may claim too many tasks and then realize that the timelines cannot be met, or the user may be assigned an important task that has a higher priority, or the users may feel that the task is simple enough that another user can complete it, and so on.

A user can only delegate personal tasks.

- The ownership of the task remains with the user who delegated the task.
- A delegated task can be delegated to only one user at a time. When a task is delegated the Delegate action becomes disabled.
- Before delegating a task, the user to whom the task is to be delegated must have the necessary permissions to work on it.
- Performing a Revoke action on a delegated task revokes the delegation first (also enables the Delegate action). The Revoke task is only available if the action is performed again, that is the Revoke action should be performed twice to revoke the claim on a delegated task.
- The Revoke action is enabled for the user to whom the task is delegated so that user is able to revoke on the delegation. Only a Manager or Admin can revoke the task.

The following information is available in the dialog box.

Name	Description
Task name	Name of the task on which the action is being performed.
Assignment type	The assignment target type: Role, Worklist, or Team.
Assigned to	The name to which the current task is assigned.
Delegated to	List of users to be delegated.
Reason For Delegation	The reason for delegating the task. This is optional.

Complete

Indicates that a task is complete.

A user with a work list manager role or a team lead role can complete executing the task on behalf of any user.

- If the selected task has follow-up tasks, a dialog box is displayed with the available follow-up tasks for the selected task.
- The application may recommend adding follow-up tasks, which can be optional or mandatory. Optional recommended follow-up tasks are selected (checked) by default. Mandatory follow-up tasks are selected (checked) and disabled. Mandatory follow-up tasks must be acted upon to process an item such as a claim, order, or service request.
- All tasks are populated with default configurations (if any), such as assignments and due date, which can be configured with more information. For more details, see the Configuration section in [Add tasks](#).

Select or clear the required follow-up tasks and click **OK** to complete the current task and initiate the selected follow-up tasks.

Note: The follow-up tasks are initiated only after the current task is completed.

Users can complete the task without selecting the follow-up tasks. If they do so, they cannot add those tasks after completing the current task.

The Complete action is not shown if there are any errors in the task user interface.

Skip

Skips tasks before they are completed. When a task is skipped, it becomes a non-workable task. The following information is displayed in the dialog box.

Name	Description
Task name	Name of the task on which this action is being performed.
Reason For Skipping	The reason for skipping the task. This is required.

The Skip action is not shown if there are any errors in the task user interface.

Filtering list results in the application

You can filter the list results shown on your home page. For example, you can restrict a list of invoices to show only invoices above a certain currency value.

If an item has subtypes, you can filter on the base item or on any of its subtypes. For example, if an item called Vehicles has a subtype called Boats that contains a subtype called Sailboats, you can filter by Vehicles, Boats, or Sailboats. Each type is shown as a check box. By default, all types are selected.

To filter list results:

1. In the **Results** panel, click the **Expand (>)** icon in the header area.
The properties used as filters are shown in an accordion display.

2. For each property that you want to use as a filter, select an operator and define the criteria for the list results.

The most common operators are shown directly below each property, however you can access all filters by clicking **Advanced**.

Note:

- If your default database search implementation does not meet the case sensitivity requirements, contact your database administrator.
- You can define advanced filters using the following wildcards in combination with the **contains** filter operator:

Wildcard character	Description
*	Use the asterisk to represent zero or more characters. For example, to find values that start with Alpha , type Alpha* .
?	Use the question mark to represent a single character. For example, to find values that have U as the second character, type ?U* .

3. Click **Apply**.

The filtered list is displayed. The filter criteria are shown at the top of the list.

The next time you run each list, it is displayed using the most recent personalized criteria.

Categorizing lists in the application

Users of your application can create categories or rename a list from the Lists panel. This is especially useful when trying to locate lists with similar names from a long set of lists.

To create a category:

1. Click the Actions menu icon and, if necessary, select **Show Customizations**.
2. Click **New Category**.
3. Type the name of the category in the text box and then click  (Save). The category name must be unique within the solution. The category appears at the end of the list.

To rename a list or category:

1. Select the list or category.
2. Click the Actions menu icon and, if necessary, select **Show Customizations**.
3. Click **Rename**.
4. In the text box, retype the name. The name of the category or list must be unique within the solution.

To include lists or categories in a new category:

1. Select the lists or categories to include in the new category. To select multiple lists and categories hold down the Ctrl key (for Windows) or the Command key (for the Mac) and click each item.
2. Click the Actions menu icon and, if necessary, select **Show Customizations**.
3. Click **New Category From Selected**.
4. Specify a unique name, and then click  (Save). The selected lists and categories (and their sub-categories) are moved to the new category.

To specify customization options:

- To remove changes you made to the default settings, click the Actions menu icon and select **Remove Customizations**. This option is available only after you have made customizations to the default lists and categories.
- To toggle between displaying and hiding your customizations to the default settings, click the Actions menu icon and select **Show Customizations** or **Hide Customizations**.
- To disable customization of lists and categories, click the Actions menu icon and select **Hide Customizations**.

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. Navigate to Available building blocks > Presentation and then select **Title**. The Title building block is added to the Added building blocks pane. The Name is supplied and cannot be changed.
3. In Configuration, select a **Type**:
 - Select **Generated by the system** to specify that the title is system-generated.
 - Select **Provided by the user** to specify that the title is entered by a user.

Note:

- Changing the Configuration > Type option changes the behavior and properties of the entity. Do 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.

- If the title is generated by the system, in **Title specification**, you can construct the title using strings and properties from the expression editor. To add a property from the expression editor, click **Insert expression**. 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.

4. 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 an Assignee building block makes the Assign, Claim, Revoke, and Forward actions available. See [Adding an assignee](#).
- Adding an Email building block automatically adds the Send Email option to the action bar for an item that is opened in the application. See [Providing email functionality](#).
- 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](#).
- Adding a Sharing building block makes the Share action available. See [Configuring security](#) for more information.
- Creating a rule that sets a property or starts a process makes the label 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 Label on the action button of the rule is Inactivate, the caption for the action button is Inactivate. See [Rule actions](#).
- 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 a multiplicity of To one).

There are two ways to make an action bar available in the application.

- When you include an Actions panel in a layout, you select the action bar to display in that panel in the application. 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.

By default, the label for each action button is shown as a tooltip in the application. See [Renaming action buttons and drop list labels](#).

To create an action bar:

1. In Workspace Documents, open the entity.
2. Navigate to Available building blocks > Presentation, and then click **Action bar**.
3. In the Add action bar dialog box, type a name for the action bar, and then click **Add**.
The action bar designer opens.
4. In the Available actions pane, select the required actions.
The selected actions are displayed in the Added actions pane.
5. In the Added actions pane, you can do the following:
 - To reorder buttons use drag and drop or click the left and right arrows.
 - To remove a button, select it and click **Remove**.
6. 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 in another group. You can group actions either into a button bar or a drop list. If you group them in a drop list, you need to provide a label for the group. The label must be unique within the action bar.

Before you begin:

- Create a new action bar or open an existing action bar and click .

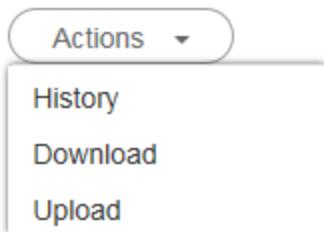
To group action buttons in a button bar:

1. In the Added actions pane, 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 in a drop list:

1. In the Added actions pane, select each action button to include in the group.
2. In Group/Ungroup, select **Group selected into drop list**.
The Drop list properties dialog box 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:

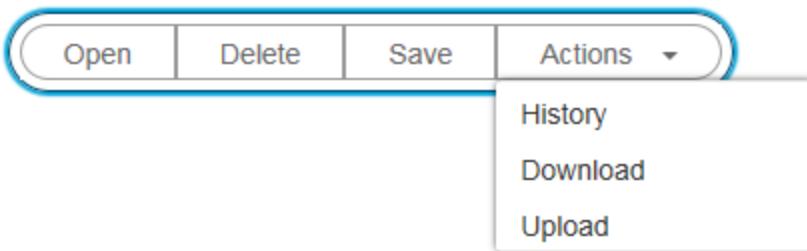


To ungroup action buttons:

1. In the Added actions pane, 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 pane, select the button bar group and the drop list group.
 2. Click **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 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 application 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:

1. Select the button, and then click **Properties**.
2. Change the label or tooltip, and then click **OK**.
The label and tooltip need not match.

To rename a drop list label:

1. Select the drop list and click **Properties**.
The Drop list label dialog box opens.
2. In **Label**, click  (Delete) to clear the existing value, type the new label, and then click **OK**.

How assignment action bars behave in the application

In the application, assignment actions behave as follows:

- If an item is assigned to a team, only the user with the lead position in the team can forward the item to a user in the team or to other groups.
- If an item is assigned to a user of a team, the assignee can forward the item to other users in the team. The lead can forward the item to other users in the team and also to other groups or group users.
- 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 an item is assigned to a specific user, only that user can forward it to other specific users.

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.

Note: Mobile apps built using the Mobile App Building Block do not support tablet devices.

This section describes adding a mobile app building block to an entity, deployment to the AppWorks Gateway, and accessing the application through the AppWorks client mobile app. The mobile app is designed to show a single entity list and layout to enable users to perform simple transactions on entity instances.

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 item (entity instance).
- Enable the display of actions within the item layout.
- Display actions from the Lifecycle building block that transition between states.

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 AppWorks Platform <release> > <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 AppWorks Platform OTDS resource ID.

4. Click **Save**.

5. Restart the OpenText AppWorks Platform service for the organization.

When using AppWorks Gateway with plain HTTP instead of HTTPS you must set the following OTDS system configuration property:

```
directory.auth.EnforceSSL = false
```

Otherwise the user will get the following error message in the AppWorks Gateway App on the mobile device:

```
{"error description" : "Must use SSL", "error" : "invalid request"}
```

Adding a Mobile App building block

To add a Mobile App building block:

1. In Workspace Documents, open the entity.
2. Type a name for the mobile app and click **Add**.
3. In the properties pane, provide a **Display name** for the mobile application.
The Display name is used as the default name of the mobile application. The name is automatically populated from step 2.

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. ■ Unauthorized - Unable to sign into the gateway. ■ Ready - The gateway is available.
List	A drop-down list of all lists within the entity.

Property	Description
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	<p>Buttons for managing the Mobile Application within the AppWorks Gateway. The following buttons are available:</p> <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 actions menu. Opening the actions 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 actions menu and presents the panel within the main mobile app screen.

Adding home pages

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.

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 home page to serve their needs. Typically, however, you will create multiple application layouts, each optimized for a different task or type of user. For example, you may need a different home page for each department in your organization.

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.

If you plan to specify a customized logo in the home page, you need to include it in your project or in a folder in your project. For example, you can add the logo image to a project folder called Images. To make the images available in your application, 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. Individual home page logos take precedence over custom themes. See [Creating a theme](#).

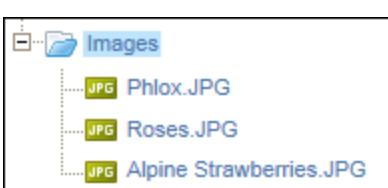
To create a folder for images (optional):

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 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**. Click **Remove** to remove the selected image or **Upload** to select a different image.
 - **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, Including multiple forms in a layout](#), and [Adding panels to a layout](#).
8. Click  (Save).

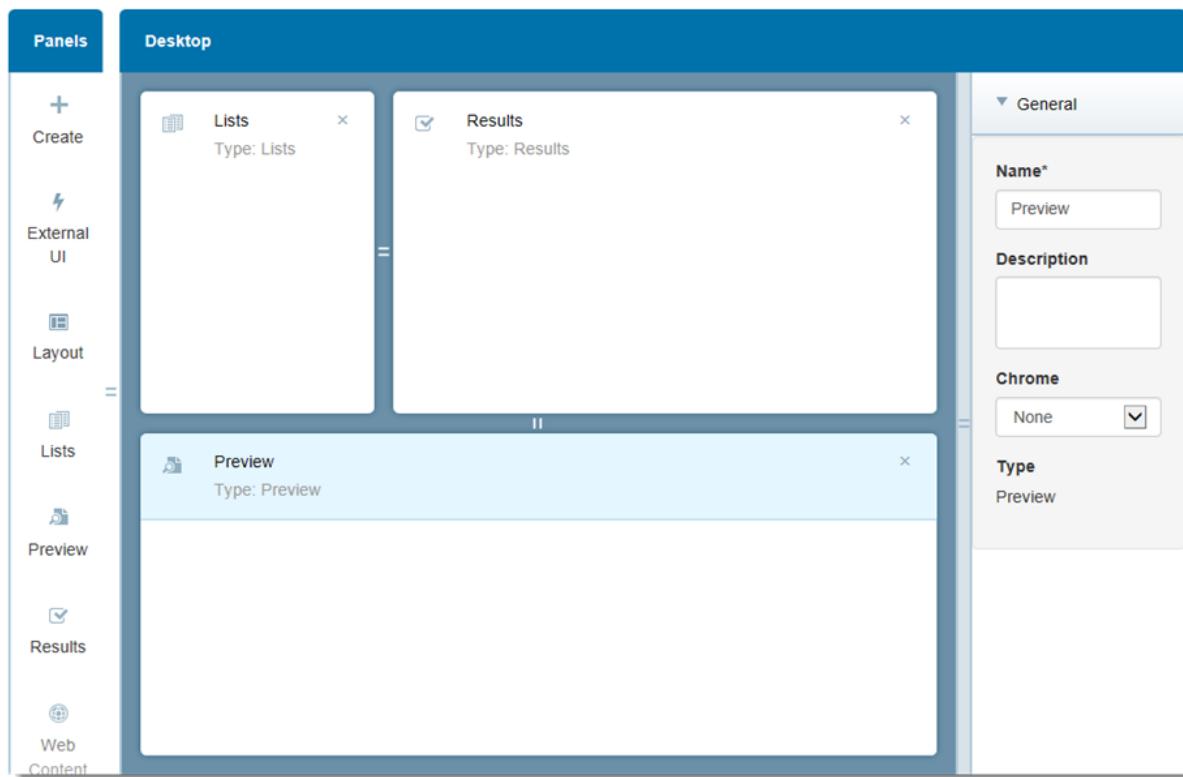
Home page example

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.

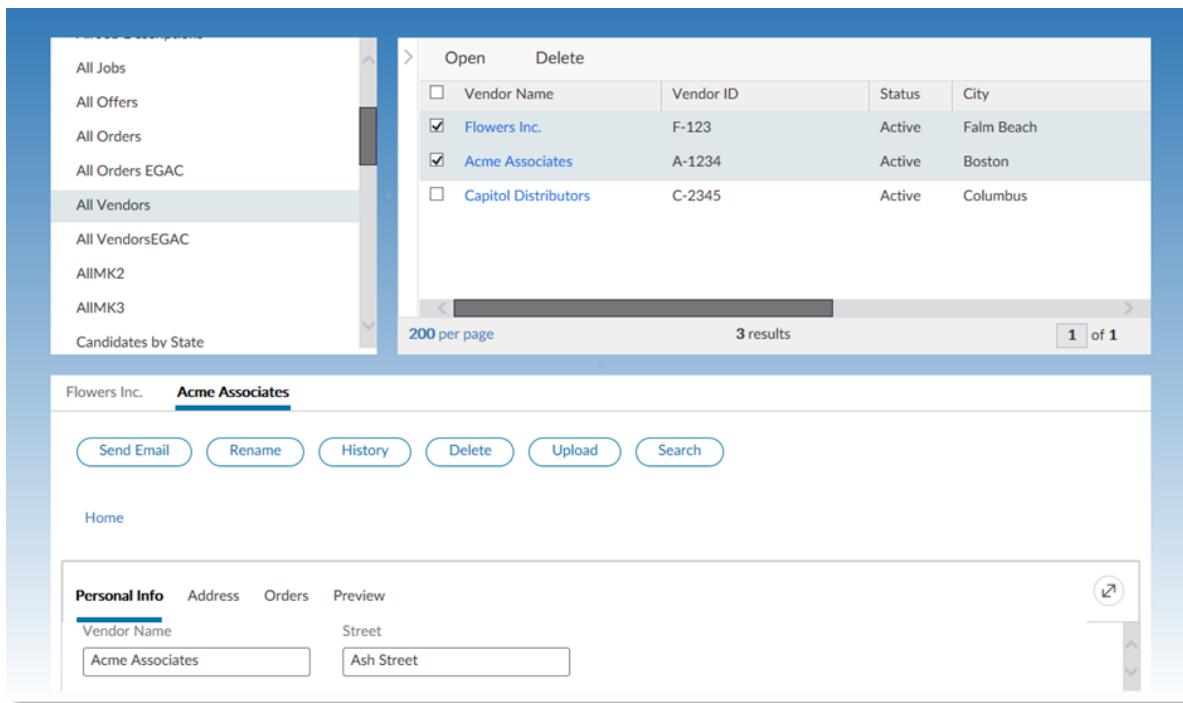
To create a home page that contains a Lists, Results, and Preview panel:

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 the application, 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.



Creating a theme

Using the Theme Editor, you can create a custom theme for a shared space or organization in your application.

A Preview option is available so that you can view the theme before publishing the project. By default, the URL is the application 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 the application 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 application 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 default configuration values for the application are shown.
3. Define the options for the theme. A header image that is specified within a home page overrides the header specified in the theme.
The recommended dimensions for the image are 31px in height and 445px in width.

4. If you want to see the theme before publishing, click **Preview**.
5. Click  (Save).
 - a. Specify the Name, Description, and Location for the theme.
 - b. Click **OK**.
6. 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 <AppWorks Platform_installdir>\<instance name>\webroot\organization\<organizationname>\themes\custom.css
 - For Linux, the custom.css file is in <AppWorks Platform_installdir>/<instance name>/webroot/organization/<organizationname>/themes/custom.css
2. Delete the class .empower-logo in the custom.css file.

Removing a custom theme

When a custom theme is created, a custom.css file is created within the AppWorks 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 <AppWorks Platform_installdir>\<instance name>\webroot\organization\<organization name>\themes\custom.css

Example for system organization:

C:\Program
Files\OpenText\AppWorksPlatform\defaultInst\webroot\organization\system\themes\custom.css

- For Linux, the file is in <AppWorks Platform_installdir>/<instance name>/webroot/organization/<organization name>/themes/custom.css

Example for system organization:

/opt/opentext/AppWorksPlatform/defaultInst/webroot/organization/system/themes/custom.css

2. Create a backup of the custom.css file and then delete it from the directory.

Chapter 7

Managing content

Content Management is provided by the File and Content List building blocks. The person who installs AppWorks 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 users create an item. They can use this option to upload a local file and display it in the application.

Note: For information about installing and configuring Content Server, OpenText Directory Services (OTDS), and other related products, see the documentation for those products, the *AppWorks Installation Guide*, and the AppWorks 16 Supported Environments.

If Content Server was selected as the document store

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 AppWorks 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 the application. For example, if an Insurance Claim entity contains the File building block, a 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 application users to add a single file to each item based on that entity. To enable 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. Navigate to Available building blocks > Functional, and then click **File**.
3. In Configuration, select one of the options for configuring the maximum size of a file that can be uploaded.

- Unlimited - can upload a file of any size. This is the default option.
 - Limited to - can set restrictions on the size of the file. Type the desired size of the file and select the metric of the file size such as MB, KB, and bytes. In the application, if the upload file size is greater than the specified size, a message is displayed and the **OK** button is disabled.
4. Select one of the restrictions for uploading a file:
 - Accept all formats - no restrictions on the file formats.
 - Accept these formats - accepts only specific formats. In the text box, type the extensions of the file formats as comma separated values. Alternately, you can click **Browse file formats**, select the required formats from the Selected file formats list, and then click **Done**.
 - Do not accept these formats - excludes specific file formats. In the text box, type the extensions of the file formats as comma separated values. Alternately, you can click **Browse file formats**, select the required formats from the Selected file formats list, and then click **Done**.
 5. Optional. In **Advanced configuration**, for Folder location, type a folder name. The folder specifies a location within the document storage area for documents belonging to this entity.
 6. Click .

Uploading and downloading files in the application

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 the application

When a user creates an item for an entity that includes the File building block, you can use the Upload button in the Create form or use the default buttons available in the action bar to upload files.

When viewing an item that includes a panel for action buttons, the default action buttons include **Upload** and **Download** options for a user to upload and download content. If the workitem does not contain content, the Download option is disabled.

A user can also display the content using the Preview panel. To achieve that, you need to add a Preview panel to the respective entity layout when designing your application.

Opening a file in an online Office editor

If the underlying document store is configured with OpenText Core, application users can open an uploaded file in a new tab in an online Office editor.

The changes made in the online Office editor are automatically saved. A new version of the file is created when the user closes the Office online tab.

Security for this action is specified in the Security building block.

Adding content

Using the Content building block, you can enable users of your application to create folders dynamically, add documents to a selected folder, and perform document management operations on a selected document.

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.

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 Security, which is typically used to secure folders. 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 the application.

To add and configure a Content building block:

1. In Workspace Documents, open the entity.
2. Navigate to Available building blocks > Functional, and then click **Content**.
The Content building block is displayed in the Added building blocks pane.

3. In Configuration, for Maximum number of files that can be uploaded, select one of the following:
 - Unlimited - user can upload any number of files. This is the default option.
 - Limited to - user can upload any number of files up to the specified maximum. Type a value greater than zero.
3. In **Advanced configuration**, accept the default folder 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>/<ItemID>/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`

Note: Configure the folder path value based on the application requirements. For example, to store documents for each entity item separately, configure a unique folder path for each item, using a unique entity property expression in the folder path value. If a unique folder path is not configured, multiple entity items might share the same path.

4. Click **Content entity configurations** to configure the added building blocks such as Content template, File, Layouts, and Title.
5. Click **Content Template > Default Template**. The Default folder is displayed under **Folder Structure**.

If necessary, you can create the folder structure as required by the application or solution. When you point to the folder name, four icons are shown:

To do this	Click this icon
Create a new sibling folder	
Add a child folder	
Delete the current folder	
Rename a folder	

8. Click  and close the Default Template window.
9. In the Added building blocks pane, click **File**.
10. In Configuration, select one of the options for configuring the maximum size of a file that can be uploaded.
 - Unlimited - can upload a file of any size.
 - Limited to - can set restrictions on the size of the file. Type the desired size of the file and select the metric of the file size such as MB, KB, and bytes. In the application, if the upload file size is greater than the specified size, a message is displayed and the **OK** button is disabled.
11. Select one of the restrictions for uploading a file:
 - Accept all formats - no restrictions on the file formats. Accepts all formats.
 - Accept these formats - accepts only specific formats. In the text box, type the extensions of the file formats as comma separated values. Alternately, you can click **Browse file formats**, select the required formats from the Selected file formats list, and then click **Done**.
 - Do not accept these formats - excludes specific file formats. In the text box, type the extensions of the file formats as comma separated values. Alternately, you can click **Browse file formats**, select the required formats from the Selected file formats list, and then click **Done**.
12. Click .
13. In the Added building blocks pane, click **Title**.
14. In Configuration, for Type, select **Provided by the user**.
15. If you want to capture custom properties for a file to be uploaded in the application, add a Create form to the child entity of the Content building block, be sure to add an Upload button, and include the required custom properties.

Important: Do not include the Content type property on a Create form that you add to the child entity of the Content building block because changing this property at run time may cause unpredictable results.

16. Optional. In the Added building blocks pane, click **Content layout**.
17. Click  (Save).

To enable the display of the content panel in the application:

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 item 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.

Working with content in the application

Users can create folders dynamically in your application. Users can upload files and perform document management operations (if they have the appropriate permissions) on folders that are either created by the user or folders available from the template. Users cannot create folders or upload files at the root folder level.

Users can delete a dynamic folder by clicking **Delete**. Users cannot delete static folders created in the application.

Deleting a folder in the Content panel deletes both the folder and the files it contains from the Content panel, but deletes only the files from the repository.

Dynamic folders inherit the permissions of their immediate parent. Files and folders present in a specific dynamic folder inherit the permissions from their immediate parent.

When users select a document, a list of operations that can be performed on it is available. The list is based on the document repository to which the document store connector is configured and on the user role and permissions. The operations are available as inline actions and also as action buttons in the panel.

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, Documentum, or OpenText Archive Center.

Application users add new documents using the + button in the Content panel and use the configured Create form to enter data. If no Create form was configured, the default upload dialog box is displayed while uploading a file.

Users may be able to perform the following operations. The operations assume that a Preview panel is provided to display the documents. If a user opens multiple documents, each one appears as a separate tab in the Preview panel so that the user can easily switch between them

- Add new documents and new versions of existing documents.
- Open documents.

- Open documents in a new tab in an online Office editor. This action is available only when the underlying document store is configured with OpenText Core. The changes made in the online Office editor are automatically saved. A new version of the document is created when the user closes the Office online tab. Security for this action is specified in the Security building block.
- Change the title of a document - renaming the title does not have any impact on the actual file name in the content repository.
- Move documents between folders.
- Drag and drop documents to a folder.
- Delete documents.
- Audit the history of documents.
- View and update document properties.
- Check out and check in documents.
- Open, download, and delete document versions.
- Zip and download documents.
- Open documents in the Brava viewer.
- Share document URLs via mail, which may require authentication on the content repository.
- Search files and folders.
- Search for documents in the repository using metadata or browse content in the repository.

Users of the application can search in the storage repository based on keywords (properties and content) and copy or link the documents that match the search criteria to an empty document. Searching is available only when users create a document without uploading any files.

To enable the application to search and attach a document in the storage repository:

1. Open the entity instance in the application.
2. Click + on the Content panel.
The **Upload Document** window opens.
3. Select the required folder in the folder list, type the document title, and then click **OK**.
An empty document is created in the Content panel with the name provided as the title.
4. Select the document and click **Search**.
The Search window opens.
5. Select the Search type, provide search text, and click **Search**.
A list of matching items is displayed.

Note: The DMS path search type does not show in the Search text box. You need to browse through the folders and select the document.

6. Select a matching item.
 - Select **Copy** to copy the content to the selected empty document.
 - Select **Link** to link the content to the selected empty document.

Note: Alternatively, you can also link the content by clicking **+** on the content panel and select **Link documents**. The search window opens. Perform step 5.

The selected item is attached to the document.

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.

Note: The Business Workspace building block cannot be customized.

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 application users to perform Content Server document management operations (such as upload, download, copy, move, and so forth) in the workspace,
- The Business Attachments panel enables application 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. Navigate to Available building blocks > Functional, and then click **Business Workspace**.
The Business Workspace building block is listed in the Added building blocks pane and a list of all building blocks is displayed in the details pane.
3. Click **Configure** and select one or more properties to store in Content Server as metadata of the workspace.
4. Click .

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 item 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 the application.
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. The order of the buttons is determined by the order in which you select them. See [Creating an action bar](#).
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 new 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 .

Using a business workspace in the application

If the business workspace feature is configured in your application, the action bar for the Results list may provide some or all of the following options:

Synchronize Workspace - If the system is configured for manual synchronization, creates the business workspace in Content Server.

If the system is configured for automatic synchronization, this button is not enabled and the workspace is created automatically in Content Server. If an error occurs during automatic synchronization the Synchronize Workspace button is enabled so that the user can manually synchronize.

Open – Opens an instance of the document template in the Business Workspace panel so that users can perform document management operations such as upload, download, copy, move, rename, share, and so forth.

To display a document template in the Business Workspace panel, the Cross-Origin Resource Sharing (CORS) plugin must be installed in the browser. Consult your system administrator if you need help.

Additional document management operations may be available inside a document template. See the Content Server documentation for details.

Open Business Workspace – Opens the Business Workspace window, where you can access business workspaces, navigate, and perform document management tasks.

The following list describes various document management operations that may be available in the application.

- To add a document, in the document template click **Add Item** (+) > **Document**, select the document, and then click **Open**.
- To download a document, select the document and then click **Download**.
- To copy a document, select the document, click **Copy**, select the destination folder in the dialog box, and then click **Copy**.
- To move a document, select the document, click **Move**, select the destination folder in the dialog box, and then click **Move**.
- To rename a document, select the document, click **Rename**, type the new name, and then click **Update**.

Additional document management operations may be available inside a document template. See the Content Server documentation for details.

Delete – Deletes the business workspace in Content Server.

Cross application workspace search – This window opens when a user clicks Synchronize Workspace for shared or related workspaces. The search window lists all matching items and the user needs to select the required item.

For automatic synchronization, the **Synchronize workspace** option is not enabled. Business workspace sharing is enabled automatically. If more than one search result is found, an error is logged and the Synchronize Workspace button is enabled for a user to link the required business workspace manually.

Click **Synchronize workspace** to view the leading system's updated data from Content Server after the business workspace is linked.

Linking an existing business workspace

You can configure the application to enable users to link an existing business workspace in Content Server to items in the application.

Synchronize Workspace - Clicking this option opens the Create or complete workspace window with the list of existing Content Server business workspaces.

To link to an item, select it and click **Complete**.

Note: Once an item is linked to a business workspace, it cannot be unlinked.

For automatic synchronization, the **Synchronize Workspace** option is not enabled. Business workspace linking is enabled automatically. If more than one search result is found, an error is logged and the **Synchronize Workspace** option is enabled for users to link the required business workspace manually.

Users can view the updated data from Content Server after the business workspace is linked, by clicking **Synchronize Workspace**.

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 the application.

Adding a message board

You can use the Discussion building block to add a message board to an entity. This facilitates collaboration among application users working on a specific 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. Navigate to Available building blocks > Functional, and then click **Discussion**.
The Discussion building block is added to the entity. No additional configuration is necessary.
3. Click  (Save).

Providing email functionality

With the Email and Email template building blocks you can provide email functionality in your applications. This enables application users to send, receive, reply to, and forward emails and attachments from a selected item. The files to be attached to the outbound mails must be added to the item through the Web service building block, Business Workspace building block, or the Content building block. Files received through inbound emails are stored in the Business Workspace building block or Content building block provided as part of the Email building block. Inbound and outbound email mailboxes must be configured in [AppWorks Administration](#).

Users can include a Business workspace building block or Content building block in the entity, and add a Content panel or Business workspace panel to the layout. When opting for a Business Workspace panel, application builders must create a unique identity for this workspace. Users upload files that they intend to send as attachments to the Content panel or the business workspace panel, depending on the builder configuration, and select them from there.

The Email building block provides an option for saving the attachments of received email to the Content building block or the Business workspace building block. This enables users to see email attachments at a single location in the Content panel or the Business workspace panel and to easily manage them using these panels. See [Adding content](#), [Adding a business workspace](#)[Adding item layouts](#), and [Adding panels to a layout](#).

Note:

- The maximum size of the attachment that can be uploaded into the respective repository depends on the system performance, repository, and connector configuration.
- To view or retrieve the created email message file, the user must refresh the browser.
- To prevent processing errors, unsupported characters in email messages are replaced with the ♦ or Ł symbols based on the database used.

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. See [Providing email functionality](#).
3. Add one or more Email templates. See [Creating an email template](#).
4. Add an email panel to a layout. See [Adding item 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	<p>Shows content (such as documents and attachments) that will be used in emails in the application. The Content building block includes the following building blocks:</p> <ul style="list-style-type: none"> ▪ Content Template ▪ File ▪ Identity ▪ Layouts - contains a Default layout ▪ Title
Email Log	<p>Maintains an audit trail of all received (inbound) and sent (outbound) emails. It is not configurable and system defined.</p>
Identity	<p>Includes 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.</p>

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 the application.

Important: When users send an email, the To: and CC: fields are limited to a maximum of 4000 characters.

To add and configure the Email building block:

1. In Workspace Documents, open the entity.
2. Navigate to Available building blocks > Functional, and then click **Email**.
3. In Configuration, for Email configuration, do one of the following:
 - If the email configuration already exists, click  and select an existing email configuration from the Select Email Configuration dialog box.
 - If you must create a new email configuration, click , and then follow the instructions on the Email Configuration page.

Note: An Email Configuration can be bound to multiple entities by using the Email Building Block within a workspace. Use [AppWorks Administration](#) to specify email configuration parameters such as From, Display name, Reply to, and Email profile User ID. See [Specifying the email configuration](#).

4. For Repository to select attachments from, select Business workspace or Content as the repository. You can also select None if you do not want to use a repository.

5. For Save email messages as files, do one of the following:

Option	Steps
Business workspace	<ul style="list-style-type: none"> a. Select Business workspace. The location to save email messages is set by default. All emails are stored as files in Emails/Messages of the business workspace. b. For Save emails as, select the required extension, .eml or .msg. When saving email messages as a .msg file, HTML formatted messages are flattened, that is, the HTML markup is removed.
Content	<ul style="list-style-type: none"> a. Select Content. b. In Save location, select the required location to save the files. You must have the Content building block added to the entity. The Content location list displays the folder paths that were created in the Content building block. c. For Save emails as, select the required extension, .eml or .msg. When saving email messages as a .msg file, HTML formatted messages are flattened, that is, the HTML markup is removed.
Do not save	Select Do not save . This will not save the emails as files.

6. For Save attachments of incoming email in, do one of the following:

Option	Steps
Business workspace	<ul style="list-style-type: none"> a. Select Business workspace. The location to save the email messages is set by default. In the application, the attachments of incoming email are stored in Emails/Attachments in the business workspace, if it is selected.
Content	<ul style="list-style-type: none"> a. Select Content. b. In Save location, select the required location to save the incoming attachments. You must have the Content building block added to the entity. The Content location list displays the folder paths that were created in the Content building block.
Do not save	Select Do not save . This will not save the attachments in the incoming emails.

7. Click **Email entity configurations**.

The Content, Email log, and Identity building blocks are displayed. Attachments to the email will be stored and managed using the Content building block. See [Adding content](#).

8. Click .

Creating an email template

Use the Email template building block to create templates that will be available for application users to send emails. You can also use email templates to send an email using web services. An email template specifies the information that will be automatically provided when a user creates an email about a selected item. Users can also select Blank email to send an email that does not use a template.

Before you begin:

- Add the Email building block to the entity. See [Providing email functionality](#).
- Create a form so that users can create items for email. See [Adding forms](#).
- Create a list so that users can select an item to associate with email. See [Adding lists](#).

To add an email template:

1. In Workspace Documents, open the entity.
2. Navigate to Available building blocks > Functional and then select **Email template**.
3. Specify the following values for the email template:

Note: To populate the values from properties, click **Insert expression**. For details, see [Using expressions](#).

- 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.

Important: The To: and CC: fields are limited to a maximum of 4000 characters

Following are some examples of information you can include in the To, Cc, Subject, and Body of an email template. When a user creates an email using the template, the values are supplied based on the item the user selected.

Item properties	{item.Properties.PropertyName}
Identity properties	{item.identity.identitytype}
System properties	{system.SystemPropertyName} See Using expressions .
User properties	{USER.Properties.PropertyName} See Using system level properties in expressions .

Properties of a related entity	{item.RelationshipName.Properties.PropertyName}
Solution variables	{config.variable} See Defining solution variables
Expressions	{3*(2+7)}
Date methods	{now}, {now.getDayOfWeek ()} See Date methods .
String methods	{equals(item.Properties.PropertyName)}, {List [item.Properties.email1, item.Properties.email2, item.Properties.email3]).toString();} See String methods .

Adding a business ID to an email template

You can add the business identifier to the subject line of an email message:

- To retain the relationship between an email and an item.
- Enable linking between an email communication and an entity instance.

Place the business identifier between [and]. You can also add it manually. The application developer can add the business identifier to the email templates in the following ways:

```
[{item.Identity.BusinessID}]  
or  
{'['+item.Identity.BusinessID+']'}
```

Linking a business ID

To process incoming email messages based on the business ID available in the subject line, link the business ID prefix to the specific entity.

To link the business ID prefix:

1. Open the email configuration in the application.
2. Add the required business ID prefix for incoming email processing.
The Email configuration page appears displaying a list of the defined business ID prefixes in your organization.
3. In the entity column, add an entity to link incoming email messages. If the business ID prefix is unique in your organization, then the entity is already selected. If your organization has multiple entities with the same business ID prefix defined, then select the entity for which the business ID in the subject line is used.

Note: If you change the business ID prefix for a running application, the new entities receive a business ID based on the new prefix, while the old entities retain their existing prefix. To manually add the business ID prefix that is no longer defined, select Add prefix.

Using Inbox lists

When you install AppWorks, a shared application called Inbox Task Management is automatically installed. The Inbox Task feature displays items from a user's AppWorks Platform Inbox in a 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.

Inbox Task Management does not support Notifications used in the MyInBox Explorer.

If someone changes the Inbox properties in AppWorks Platform after the Task Management application is installed, the changes are not reflected in the application.

To make the Inbox lists available in the application, you need to set solution security for the Inbox Task Management application. See [Defining security for Inbox Task Management](#).

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 AppWorks 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 AppWorks Platform installation.
- Ensure that you have an MS-SQL Server, Oracle, or PostgreSQL database to hold the processed emails.

Creating a database configuration

Note: You need to create a database configuration only if you do not intend to use the AppWorks 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.
See *JDBC Details Interface* in the *AppWorks Platform Advanced Development* documentation.

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* in the *AppWorks Platform Advanced Development* documentation.
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.

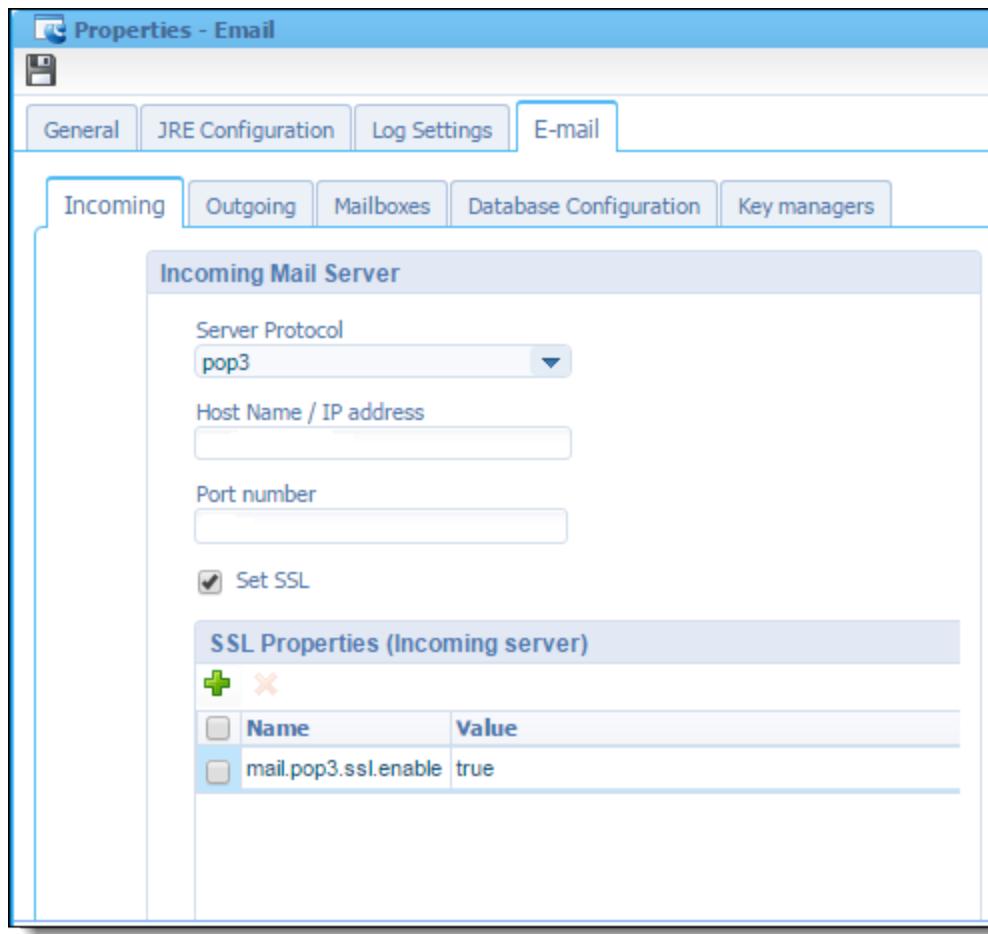
Configuring the E-mail connector

Configuring the E-mail connector involves the following configurations:

- Inbound
- Outbound
- Mailboxes
- Database
- Key managers

To configure the E-mail connector:

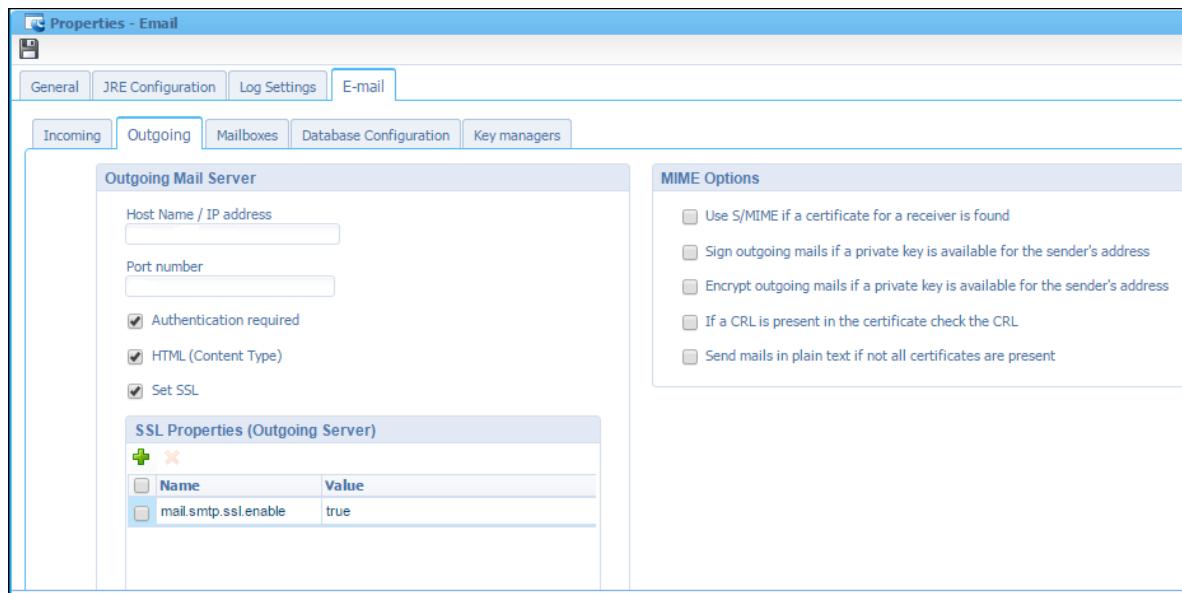
1. On the AppWorks Platform Welcome page, click  (System Resource Manager).
The Email service container is displayed in the Email service group. These are created automatically when AppWorks Platform is installed.
2. In the Service Containers pane, right-click the E-mail service container, and then click **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.
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.

- 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	Whether S/MIME is enabled.

Option	Description
receiver is found	
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.

7. 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 AppWorks 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>

```

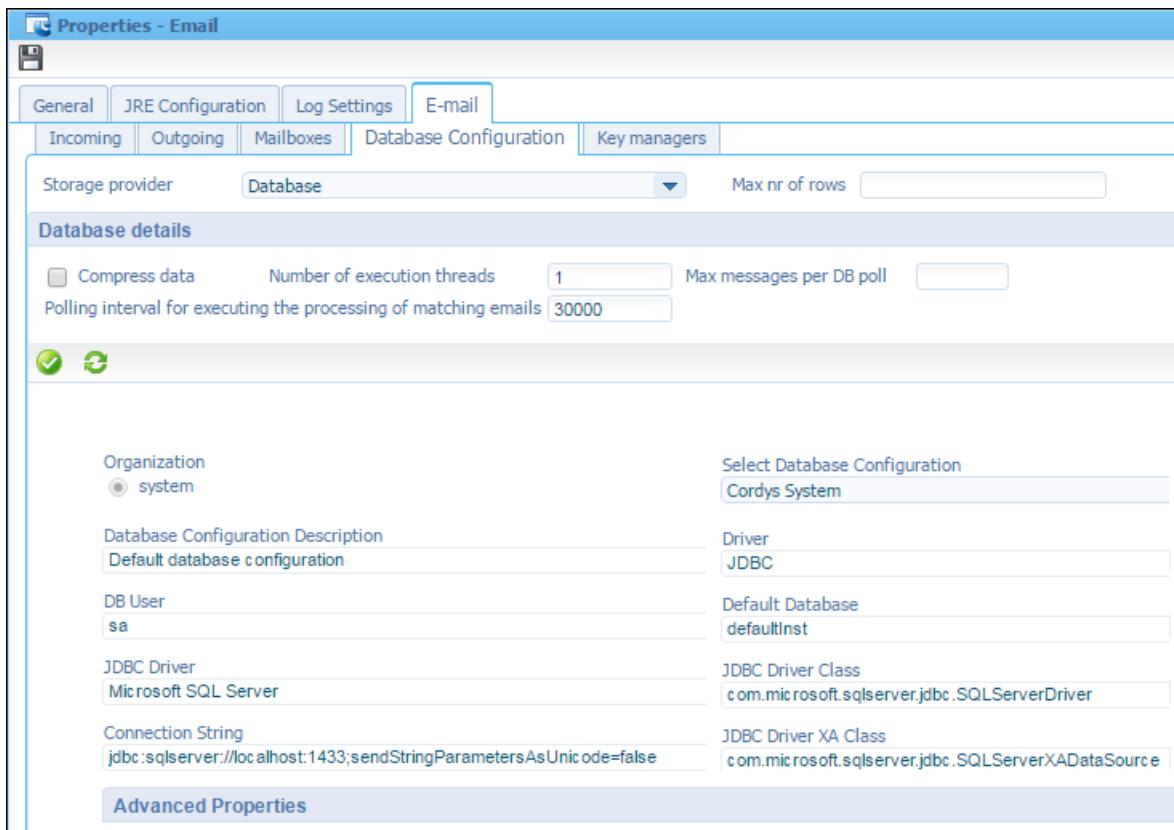
Customize the following parameters in the XML configuration for your email account:

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 AppWorks Platform is installed on the server. In o=vanenburg.com, replace vanenburg.com with the domain name.

Note: Provide a superuser/user that has permissions to create and upload documents.
In the following example, SYSTEM is the superuser:

```
cn=SYSTEM,cn=organizational  
users,o=system,cn=cordys,cn=defaultInst,o=opentext.net
```

8. On the Database Configuration tab, enter the following information.



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.

9. 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. <p>Note: You might have created a database in the previous section, Creating a database configuration.</p> <ul style="list-style-type: none"> 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 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> <ul style="list-style-type: none"> a. In the Select Database Configuration list, select New Database Configuration. The New Database Configuration window opens. b. For Name and Description, enter the name and description of the database configuration. The description is optional. c. Enter the relevant details in the corresponding fields. <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.



10. 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.

11. 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 specific 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 [Adding building blocks to entities](#).

Important: There is a difference between customizing an entity and [Using subtyping for advanced modeling](#). 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 *AppWorks Platform Advanced Development* documentation.

Customizing platform packages

AppWorks Platform also provides entities that can be used like Model contracts or Model packages. These packages are already installed and imported during AppWorks 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.

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.

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 AppWorks 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.

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 **Enable reuse in other applications**.
3. Click **OK**.
4. In the Details pane for an existing building block, select **Enable reuse in other applications**.

If **Enable reuse in other applications** is not selected for an entity, the **Enable reuse in other applications** 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 **Enable reuse in other applications** 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 AppWorks 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.

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.

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.

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.

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:

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**.

Note:

- 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.
 - If the package is in use, it is not removed.
 - If the package is not in use, the removal is complete when the application package disappears from the Imported Application Packages section.

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.

Using subtyping for advanced modeling

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, application 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. 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 specific 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 the AppWorks Platform Advanced Development documentation.

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. Select the building block to be replaced.

Note: If the building block is replaceable, a **Replace** icon is shown in the highlighted bar around the building block.

3. Click **Replace**.

Note: If there is an option to start with a copy from the original building block, a confirmation dialog box opens. Click **Yes** to start with a copy of the original or **No** to start from scratch.

4. Open the building block designer and 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

Note: For a customized or subtyped entity, it is possible to replace inherited activity flows only and new activity flows cannot be added.

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 planning considerations for applying subtyping within your business domain. The insurance example described previously in this topic is 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.

Determine whether the supertype will 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.

Support for large tables in MS SQL

You can provide support for a base entity that is expected to contain a large number of subtypes with properties that exceed the 1024 column limit for a table.

Note: This is used only for MS SQL and needed only when you have subtypes where the aggregate number of properties exceeds 1024.

To provide support for large tables:

1. In the `wcp.properties` file, specify the base entity in the property `com.opentext.entityruntime.entity.requires.wide.table`.
The value of the property is `<packagename>.<entityname>`.
2. Resolve the `packagename` and `entityname`.

Assume that there is a workspace in CWS named WideTable Workspace with a project named MyBaseEntityProject. In this project, an entity named MyBaseEntity is the base entity from which a subtype is derived.

If you select the project > **Packaging -> Package Properties** the Package Name is My Company Wide Table Project. The value of the property is My Company Wide Table Project.MyBaseEntity.

When you publish or deploy an application with an entity defined in the property all of the properties defined in the subtypes are converted to a sparse column (marked with the Sparse attribute) and a column set is created to manage all space columns automatically (`S_COLUMN_SET(Column Set, XML(,),null)`).

The change takes effect only when the data model has been changed for the application that is reflected when you add a new property or a new subtype. If you publish or deploy an existing application without any changes to the data model, existing columns are not changed.

Chapter 10

Using the Identity Package

Open Text Directory Services (OTDS) handles user management and authentication for AppWorks. The Identity Package provides a set of predefined entities that are automatically installed with AppWorks so that you can use them in your applications. For example, by creating a relationship to the Person entity, you can include information about different people in your forms, lists, and so forth.

Important: The Identity Package requires that you use Open Text Directory Services (OTDS) and the OTDS Push Connector.

Using the Identity package, you can:

- Manage organizational model master data in the application. Users with the appropriate privileges can create the organizational structure or model, such as organizational units, worklists, business contacts, countries, and more..
- Assign users to organizational units 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 the application 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 AppWorks.

Notes:

- If you managed your organizational model using the AppWorks Platform Organization Model in the past, now you will manage the model in the application with the Identity Package.
- If you build entity-based applications, it is required that you use OTDS to facilitate user management. If you have AppWorks 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

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 the application in OTDS. See the *AppWorks Platform* documentation.
- Make the Identity Package available in your application 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 your 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 use 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).

Entities provided in the Identity Package

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 the application. 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 AppWorks 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 the application and view them on the Identity home page. Management of these identities is performed from the Identity Homepage in the application.

- [Address](#)
- [Assignment](#)
- [Country](#)
- [County](#)
- [EmailAddress](#)
- [EmailAddressType](#)
- [EmergencyContact](#)
- [EmergencyContactRelationship](#)
- [Enterprise](#)
- [Genders](#)
- [Identity](#)
- [Language](#)
- [Organizational unit](#)
- [NationalId](#)
- [Person](#)
- [Phone Number](#)
- [PhoneNumberType](#)
- [Position](#)
- [SocialAccount](#)
- [SocialMedia](#)
- [State](#)
- [Title](#)

- [VisaType](#)
- [Worklist](#)

Address

The Address entity contains the following building blocks.

Lists

Name	Label	Description	Attributes
AllAddresses	All addresses	A list of all available addresses.	Address line 2 City Note Postal code Street address 1
AllAddressesInternal	All addresses internal	An internal list that is used by the Identity package and that should not be replaced	ID ItemId City Note Postal code Street address 1
SelectDefaultAddress	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	Label	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

Property	Label	Description	Data type	Length
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	Label	Description	Type	Multiplicity	Related entity	Make bidirectional
County	County	Enables adding a county to an address.	To peer	To one	County	Unidirectional
ToCountry	To country	Enables adding a country to an address.	To peer	To one	Country	ToAddress of Country
ToState	To state	Enables adding a state to an address based on a selected country.	To peer	To one	State	ToAddress of State

Assignment

The Assignment entity contains the following building blocks.

Lists

Name	Label	Description	Attributes
AssignmentListInternal	Assignment list internal	Internal list of assignments.	End date Principal Start date ItemId

Properties

Property	Label	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	Label	Description	Related entity	Type	Multiplicity	Make this relationship bidirectional
toPersonToOne	To person to one	Enables linking person(s) with assignments.	Person	To peer	To one	ToAssignment
toPositionToOne	To position to one	Enables adding assignments to an available position.	Position	To peer	To one	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	Label	Description
Create	Create	Used to create a country.
Default	Default	Used to display information about a country.

Layouts

Name	Label	Type	Forms used
Default_Layout	Default Layout	Full	Default
Preview	Preview	Preview	Default

Lists

Name	Label	Description	Attributes
AllCountries	All countries	List of available countries used to associate a country with an address.	Address line 2 City Note Postal code Address line 1
CountryListInternal	Country list internal	An internal list that is used by Identity package and that should not be replaced.	ID ItemId Country code Name

Properties

Name	Label	Description	Data type	Length
Country_Name	Name	The name of the country.	Text	64
ISO_country_code	Country code	The ISO country code.	Text	2

Relationships

Name	Label	Description	Related entity	Type	Multiplicity	Make this relationship bidirectional
State	State	Country can have no or more than one State as child.	State	To child	To many	ToCountry of Address
ToAddress	To Address	Address will have country information.	Address	To peer	To one	ToCountry of Address

County

The County entity is a child of the State entity.

Lists

Name	Label	Description	Attributes
AllCounties	All counties	Default list that shows all the available counties	Name

Properties

Name	Label	Description	Data type	Length
Name	Name	Name of the county	Text	128

EmailAddress

The EmailAddress entity contains the following building blocks.

Lists

Name	Label	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	Label	Description	Data type	Length
Address	Address	Email address.	Text	128

Relationships

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
Type	Type	Required to define the type for an email address	To peer	To one	EmailAddressType	-none-

EmailAddressType

EmailAddressType is a master data entity that can be created by using the Identity package.

Lists

Name	Label	Description	Attributes
AllEmailAddressTypes	All email addresses	List of available email address types.	Type

Properties

Name	Description	Data type	Length
Type	Type of email address.	Text	128

EmergencyContact

EmergencyContact is a child of the Person entity.

Lists

Name	Label	Description	Attributes
AllEmergencyContacts	All emergency contacts	Default list that shows all the emergency contacts.	Relationship

Properties

Name	Label	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	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
Person	Person	This is a relationship with a parent Person.	To peer	To one	Person	-none-
Relationship	Relationship	This is used to add an emergency contact for the person and define the contact's relationship	To peer	To one	EmergencyContactRelationship	-none-

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
		(spouse, mother, father, and so forth).				

EmergencyContactRelationship

The EmergencyContactRelationship entity serves as a pick list. Entries will be things such as Mother, Father, Grandmother, and so forth.

Forms

Name	Label	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	Label	Description	Attributes
AllEmergencyContactRelationships	All emergency contact relationships	List of all available emergency contact relationships.	Relationship

Properties

Name	Label	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.

See [Organizational unit](#) for a list of building blocks.

Lists

Name	Label	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	Label	Description
Create	Create	Used to create a gender.
Default	Default	Used to display information about a selected gender.
Preview_Form	Preview Form	

Layouts

Name	Label	Type	Forms used
Preview	Preview	Preview, Full	Preview Form

Lists

Name	Label	Description	Attributes
Genders_List	All genders	List of all available genders.	Gender
GendersListInternal	Genders list internal	An internal list that is used by Identity package and that should not be replaced.	Id ItemId Gender

Properties

Name	Label	Description	Data type	Length
Gender	Gender	The name of the gender.	Text	64

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	Label	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	Label	Description	Attributes
AllGroupsListInternal	All groups list internal	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	Label	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	Label	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	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
GroupToIdentity	Group to identity	Enables adding a user to a group.	To peer	To many	Identity	IdentityToGroup

Identity

The Identity entity is the base entity of the Identity Package. Its properties are used by other entities in the package.

Lists

Name	Label	Description	Attributes
GroupList	All groups	List of the available groups.	Name Description EntityType
OrganizationalUnitList	All organizational units	List of the available organizational units.	Name Description EntityType
UsersList	All users	Default list of the available users.	Name Description EntityType

Properties

Name	Label	Description	Data type	Length
Description	Description		Text	256
IdentityDisplayName	Identity display name		Text	256
Name	Name		Text	256

Relationships

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
IdentityToGroup	Identity to group	Enables getting the list of groups of which the user is a member.	To peer	To many	Group	GroupToIdentity
IdentityToRole	Identity to role	Enables getting the list of roles to which the user was added as a member.	To peer	To many	Role	RoleToIdentity
IdentityToWorklist	Identity to worklist	Enables getting the list of worklists to which the user was added as a member.	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	Label	Description	Attributes
AllLanguages	All languages	The list of available languages.	Name

Properties

Name	Label	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	Label	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	Label	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	Label	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	Label	Description	Attributes
OrganizationListInternal	Organization list internal	An internal list of organizational units. Not visible to users. Should not be replaced.	Name
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	Label	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	Type	The type of the organizational unit: Hierarchical	Enumerated Text	64

Relationships

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
Addresses	Address es		To peeler	To many	Address	-none-
DefaultAddress	Default Address	Enables associating an address with a unit.	To peeler	To one	Address	-none-
EmailAddresses	Email address es	Enables associating email addresses with a unit.	To peeler	To many	EmailAddress	-none-
PhoneNumbers	Phone numbers	Enables associating phone numbers with a unit.	To peeler	To many	PhoneNumber	-none-

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
Position	Position	Enables adding positions to a unit.	To child	To many	Position	ToOrganizationalUnit
PrimaryEmailAddress	Primary email address	Enables associating an email address with a unit.	To peer	To one	EmailAddress	-none-
PrimaryPhoneNumber	Primary phone number	Enables associating a phone number with a unit.	To peer	To one	PhoneNumber	-none-
SocialAccounts	Social accounts	Enables associating social accounts with a unit.	To peer	To many	SocialAccount	-none-
SubUnits	Sub units	Enables associating units with a unit as 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

Name	Rule summary
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 AppWorks. 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.
SummaryForm	Used to maintain summary information about a person.

Layouts

Name	Label	Type	Forms used
DefaultLayout	DefaultLayout	Full	HeaderInformation SummaryForm
UserLayout	Preview	Preview	Create

Lists

Name	Label	Description	Attributes
AllPersons	All persons	A list of available persons.	

Name	Label	Description	Attributes
PersonList	All contacts	A list from which a user can select a person.	Display Name Email Is internal
PersonListInternal	Person list internal	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	Label	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	Label	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	Is internal		Boolean	
LastName	Last name	The person's last name.	Text	64
MaritalStatus	Martial status	The person's marital status (Single, Married, Widow, Divorced).	Enumerated Text	
MaritalStatusSince	Marital status since	The date when the person's marital	Date	

Name	Label	Description	Data type	Length
		status became effective.		
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	32
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	32
physicalDeliveryOfficeName	Office	The name of the office where deliveries can be made for the person.	Text	64
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	Enumerated Text	

Name	Label	Description	Data type	Length
		(none, Mr, Ms, Mrs).		
SecondLastName	Second last name	The person's second last name.	Text	128
Suffix	Suffix	The suffix for the person's name. This property was deprecated in 16.2.1.	Text	16
title	Job title	The person's title.	Text	64
User_ID	User ID	The person's user ID. This property was deprecated in 16.2.1.	Text	64

Relationships

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
Addresses	Addresses	Enables adding addresses to a person.	To peer	To many	Address	-none-
CountryOfBirth	Country of birth	Enables adding a country to a person as the country of birth.	To peer	To one	Country	-none-
EmailAddresses	Email addresses	Enables adding email addresses to a person.	To peer	To many	EmailAddress	-none-
EmergencyContact	Emergency contacts	Enables adding a contact as an	To child	To many	EmergencyContact	-none-

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
		emergency contact of a person.				
NationalId	National IDs	Enables adding national ids to a person.	To child	To many	NationalId	-none-
PersonToGender	PersonToGender	Enables adding gender information for a person.	To peer	To one	Genders	-none-
PersonToUser	PersonToUser	Enables linking a user to a person.	To peer	To one	User	toPerson
PhoneNumbers	Phone numbers	Enables adding phone numbers to a person.	To peer	To many	PhoneNumber	-none-
PreferredLanguage	Preferred language	Enables adding a language as a preferred language	To peer	To one	Language	-none-
PrimaryEmailAddress	Primary email address	Enables adding an email address as the primary email address	To peer	To one	EmailAddress	-none-
PrimaryEmergencyContact	Primary emergency contact	Enables adding an	To peer	To one	EmergencyContact	-none-

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
		emergency contact as the primary emergency contact.				
PrimaryPhoneNumber	Primary phone number	Enables adding a phone number as the primary phone number.	To peer	To one	PhoneNumber	-none-
RegionOfBirth	Region of birth	Enables adding the state as the region of birth.	To peer	To one	State	-none-
SecondTitle	Second title	Enables adding a title as a second title.	To peer	To one	Title	-none
SocialAccounts	Social accounts	Enables adding social accounts.	To peer	To many	SocialAccount	-none-
ToAssignment	ToAssignment	Enables assigning a person to any position.	To peer	To many	Assignment	toPersonToOne
VisaType	Visa type	Enables adding visa type information.	To peer	To one	VisaType	-none-
WorkAddress	Work Address	Enables adding any address of	To peer	To one	Address	-none-

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
		the person as a work address.				

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	Label	Description	Attributes
AllPhoneNumbers	AllPhoneNumbers	A list of available phone numbers.	Extension Number

Properties

Name	Label	Description	Data type	Length
Extension	Extension	Extension of the phone number	Text	64
Number	Number	Phone number	Text	64

Relationships

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
Type	Type	Enables adding the type of the phone number.	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	Label	Description	Attributes
AllPhoneNumberTypes		A list of available phone number types.	Type

Properties

Name	Label	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	Label	Description	Attributes
PositionList		A list of positions.	
PositionListInternal		A list of positions for internal use.	

Properties

Name	Label	Description	Data type	Length
Description	Description	A description of the position.	Text	64
Lead	Lead	Whether this is a lead position	Boolean (True or False)	
PositionName	Position name	The name of the position.	Text	64

Relationships

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
ToAssignment	ToAssignment	Enables adding an assignment to a position.	To peer	To many	Assignment	toPositionToOne
ToOrganizationa lUnit	ToOrganizationa lUnit	Enables adding positions under an organizational unit.	To parent	To one	Organizational Unit	Position
ToWorklist	ToWorklist	Enables adding positions to a work list.	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	Label	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	Label	Description	Attributes
AllRolesInternal	All roles internal	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	Label	Description	Data type	Length
Description	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

Name	Label	Description	Data type	Length
Package_Name	Package name		Text	64

Relationships

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
RoleToIdentity	Role to identity	Enables getting the members of a role.	To peer	To many	Identity	IdentityToRole

SocialAccount

The SocialAccount entity provides the following building blocks.

Lists

Name	Label	Description	Attributes
AllSocialAccounts	All social accounts	A list of all social accounts	Name

Properties

Name	Label	Description	Data type	Length
InstantMessagingId	Instant Messaging ID	ID on the selected social media.	Text	128
URL	URL		Text	256

Relationships

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
Domain	Domain	Enables adding the domain information to social media.	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	Label	Description	Attributes
AllSocialMedia	All social medias	List of all social media.	Name

Properties

Name	Label	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	Label	Description	Attributes
AllStates	All states	List of states	Name ID ID1
AllStatesInternal	All states internal	Internal list of states	

Properties

Property	Description
Name	The name of the state.

Relationships

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
County	County	Enables adding a county to a state.	To child	To many	County	-none-
ToAddress	To address	Enables adding a county under an	To peer	To many	Address	ToState

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
		address.				
ToCountry	To country	Enables adding a county to a country.	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	Label	Description	Attributes
AllTitles	All titles	Lists the available titles.	Name

Properties

Name	Label	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	Label	Type	Forms used
DefaultLayout	Default layout	Full	HeaderInformation((For Header) SummaryForm (for Summary tab) RolesForm (for Roles tab) GroupsForm (for Groups tab)
User_Preview_Layout	Preview	Preview	Preview

Lists

Name	Label	Description	Attributes
AllUsers	All users	A list of available users.	Full Name User Id
AllUsersInternal	All users internal	An internal list that is used by Identity package and that should not be replaced.	Display Name Email Is Internal

Properties

Name	Label	Description	Data type	Length
Description	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	Text	64

Name	Label	Description	Data type	Length
		component.		
Name	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	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
Personalization	Personalization		To child	To many	Personalization	-none-
toPerson	To person	Enables associating a user with a person.	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	Label	Description	Attributes
AllVisaTypes	All visa types	A list of available visa types.	Name

Properties

Name	Label	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	Label	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	Label	Description	Attributes
AllWorklistInternal		An internal list that is used by the Identity package and	Description

Name	Label	Description	Attributes
		that should not be replaced.	
Worklists	All worlists	A list of available worklists.	Id ItemId Description Worklist name

Properties

Name	Label	Description	Data type	Length
Description	Description	A description of the worklist.	Text	64
WorkListName	WorkListName	The name of the worklist.	Text	128

Relationships

Name	Label	Description	Type	Multiplicity	Related entity	Make this relationship bidirectional
ToPosition	ToPosition	Enables adding a position in a worklist.	To peer	To many	Position	ToWorklist
WorklistToIdentity	WorklistToIdentity	Enables associating a person with a worklist,	To peer	To many	Identity	IdentityToWorklist

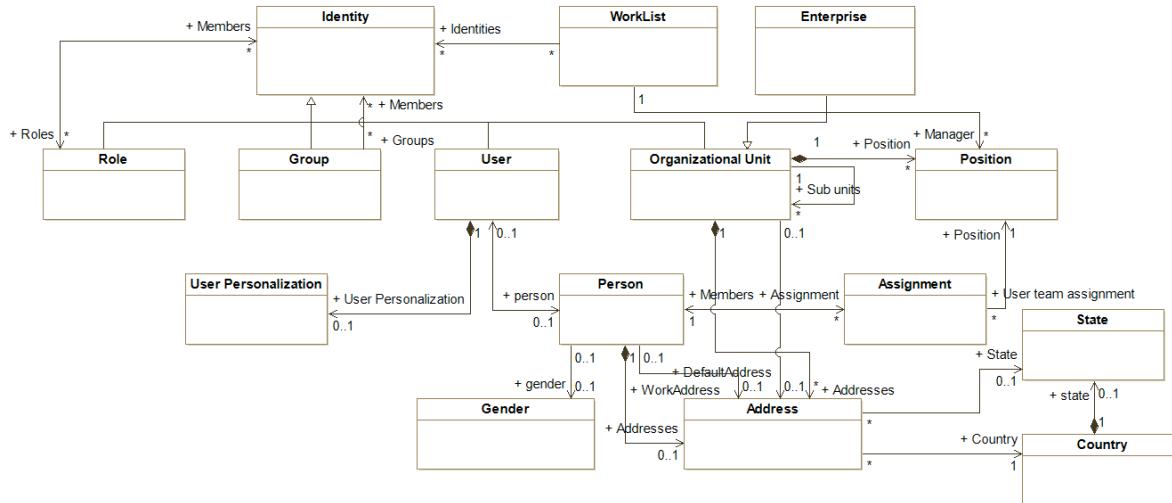
Rules

Name	Rule summary
OnCreate	Start Process BPMs/CreateWorklistEntryInNotificationRepository When a new item is created
onDelete	Start process BPMs/DeleteWorklistEntryInNotificationRepository When an item is deleted
WorklistNameMandatory	Show error If item.Properties.WorkListName is null When a new item is created

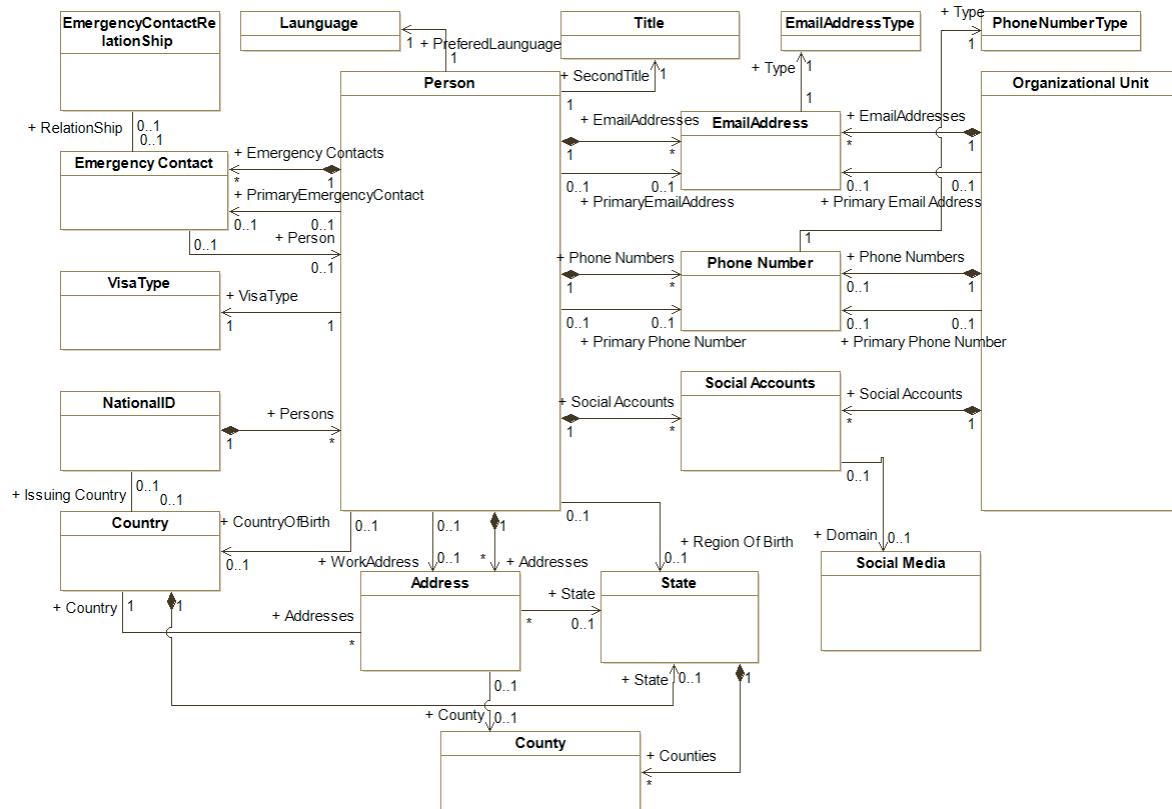
Identity domain model

The Identity component domain model is split into two diagrams - one showing an overview of the domain model (without details of the Person entity) and another one showing a more detailed diagram of the Person entity.

Overview diagram



Detailed diagram of Person entity



Identity roles

The Identity package provides the following roles:

- Identity Administrator
- Identity Push Service
- Identity User

Using OTDS with the Identity Package

The following sections describe how OTDS is used with the Identity Package.

Information pushed from OTDS to AppWorks 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 OpenText CARS, entity modeling, and the application user interface. Synchronization from OTDS to AppWorks Platform is one-way. The OTDS Push connector will only synchronize information from OTDS to AppWorks Platform. That is, you cannot add users in the application and synchronize them with OTDS.

See [Managing identities](#).

Managing information pushed to OTDS

Data is only pushed automatically from OTDS to AppWorks Platform and not from AppWorks Platform to OTDS. There are situations when data must be synchronized from AppWorks Platform to OTDS. In these situations, you need to use the OTDS Push roles functionality available in Security Administration. For more information, see the [Synchronizing roles from AppWorks Platform to OTDS](#) topic in the *AppWorks Platform Administration Guide*.

The situations when you need to run the OTDS Push Roles are when a functional or application package role is created in OpenText CARS. That is the case when:

- A package with a functional or application role is deployed in AppWorks Platform.
- An updated package is deployed and the package has new, renamed, or updated roles. A role is updated when either the role or role membership is changed.
- AppWorks Platform is upgraded to a newer version or fix pack, which might contain new or updated roles.
- A package is undeployed from AppWorks Platform.
- A CWS project containing roles is published.

Note: The Push roles to OTDS functionality cannot detect a role name change. Therefore, when a role is renamed, the user and group membership for that role is lost. You must map the role to the user and group membership again.

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
displayName Note: The default value is __NAME__.	Identity	IdentityDisplayName
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

OTDS Attribute	Entity to which it is mapped	Entity modeling property name
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 outName	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.

You can add customized properties to a person entity, which are listed in OTDS as dynamic attributes. These dynamic attributes can be mapped to the existing OTDS attributes.

Important: If you customize the person entity in order to push new properties to OTDS, you need to edit the Security building block of the person entity and provide the Identity Push Service role for the newly created properties.

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 expressions

The expression editor is a tool that enables you to create expressions for making logic decisions and calculations based on the data in your application.

The Basic mode is easy to use, while the Advanced mode is more powerful and enables you to express more complex conditions. You can design filters in the Basic mode and, if necessary, switch to the Advanced mode to edit expressions in the text mode. The expression editor is used by multiple building blocks.

This topic describes:

- Using the expression in the Basic mode. See [Selecting a property](#)
- Using the expression editor in the Advanced mode. See [Creating an advanced expression](#)

Selecting a property

Building blocks provide the option to select a property from the available list, using the **More properties** option.

To select a property:

Select a property from the list and click **Select**.

The property becomes available.

More properties

If the required property is not listed, select **More properties**.

The Select property dialog box opens. It contains:

- Properties related to the selected entity.
- Properties from the building blocks.
- Optional: User properties.

Note: Properties of the data type Image are not displayed in the properties list.

Creating a basic expression

The 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. For details on selecting a property, see [Selecting a property](#).
- Use Plus **+** and Minus **-** to add and remove conditions.

Creating an advanced expression

Important: Some programming knowledge is required to include the information described in this topic in your application.

Use the expression editor to build an expression using one or more properties or functions. The expression editor is available in the Condition > Advanced mode section for multiple building blocks.

This expression editor enables you to insert a function or a property as a full text expression, for example `{item.Properties.MyProp}`.

To build an expression:

1. Create an entity and add a building block.
2. Click **Open expression editor** or **Insert expression**.
The Expression editor dialog box opens.

Note:

- **Open expression editor** is used to build expressions with properties, whereas **Insert expression** adds the selected property to the string.
- The Expression editor dialog box contains:
 - Properties related to the selected entity.
 - Properties from the building blocks.
 - Properties from the related entities. For a list of aggregation operations, see [Arrays and aggregation](#).
 - User properties. The user entity provides information about the current user.

3. Select the required properties to build your expression and click **Insert expression**.
This expression is copied into the Expression editor dialog box.

Using system level properties in expressions

System level properties are available for use in expressions. 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 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.

Using durations to specify relative dates and times

Durations provide the capability 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 expression 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 expression 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 1-4 integers, each integer representing one of the following units: [, Months [, Days[, Hours[, Minutes]]]]. For example, three integers represent Months, Days, and Hours.

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 2 months, 3 days, where the omitted hours, and minutes are all assumed to be 0.

```
Show warning "Too soon!"
```

```
If item.Properties.EndDate < item.Properties.StartDate + duration(2, 3)
```

This example is, once again, an indirect comparison of the independently invalid "If Duration is less than 2 months 3 days".

Using 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.

An 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
<code>3 * (2 + 7)</code>	Simple integer arithmetic with parenthesis
<code>75.3 * (ivar + 7)</code>	Decimal numbers and variables, with type coercion for mixed number types
<code>ivar == 3 "word" != svar</code>	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: <code>ivar == 3 OR "word" != svar</code>
<code>substring("string", 2, 6) + "test"</code>	A variety of functions and methods are supported using the standard function(argument, argument, ...) syntax
<code>length(svar+'test') * 7</code>	Function arguments may be expressions
<code>true && 'true'</code>	Automatic type coercions (this example shows a string coerced into a boolean value)
<code>now + 1 end of months + 2 mondays</code>	Complex date and time calculations (this example computes the 2nd monday of next month).
<code>(ivar>=37 ? ivar+77 : (ivar+77)%37) * 17 < 44</code>	The result of a trinary "if" operator may be used as an expression term (note the parenthesis)

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
<code>value = 2034183616.0 / 100000</code>	20341.83616 (as expected)
<code>value = 2034183616 / 100000</code>	20341 (this is actually integer division followed with a cast to Decimal)

Expression	Result
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.

round

This function returns the rounded value and accepts two parameters. The first parameter can be of type float, decimal, currency, or integer. The second parameter is optional, and is of type integer. Any null values are ignored.

Function	Arg	Type	Returns
round(arg, multiple)	arg	float, decimal, currency, integer	Returns the rounded value.
	multiple (optional)	integer	

```
round = round(item.Properties.Property)
round = round(item.Properties.Property1, item.Properties.Property2)
```

floor

This function returns the floor value and accepts two parameters. The first parameter can be of type float, decimal, currency, or integer. The second parameter is optional, and is of type integer. Any null values are ignored.

Function	Arg	Type	Returns
floor (arg, multiple)	arg	float, decimal, currency, integer	Returns the floor value.
	multiple (optional)	integer	

```
floor = floor(item.Properties.PropertyName)
floor = floor(item.Properties.Property1, item.Properties.Property2)
```

ceil

This function returns the ceil value and accepts two parameters. The first parameter can be of type float, decimal, currency, or integer. The second parameter is optional, and is of type integer. Any null values are ignored.

Function	Arg	Type	Returns
ceil (arg, multiple)	arg	float, decimal, currency, integer	Returns the ceil value.
	multiple (optional)	integer	

```
ceil = ceil(item.Properties.PropertyName)
ceil = ceil(item.Properties.Property1, item.Properties.Property2)
```

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 the application, 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.
isInOrganizationalUnit(unit-name)	Returns true if the current user is a member of the organizational unit, false otherwise.
GetEntityTypeById(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.
GetEntityTypeByName(entity-id, solution-name)	Returns the entity's name given its ID and solution name.

Expressions on task states

Sometimes you'll want to create an expression that is conditional on the state of a task (typically in a Task entity). When writing such an expression you'll need to use the internal value of the task state, as shown in the following example.

```
// If the task is not completed or obsolete ... item.Task.State != 5 &&
item.Task.State != 8
```

The following table shows the task state internal values.

State	Internal Value
Created	1

State	Internal Value
Assigned	2
In Progress	3
Suspended	4
Completed	5
Failed	6
Paused	7
Obsolete	8
Error	9
Internal Pause	10

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
date	create a date	date(arg1 ...)	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,

Operator	Description	Usage example	Data type(s)
Keywords and functions			
			minute) ■ date(year, month, day, hour, minute, second)
time	create a time	time(arg1, ..., argn)	All arguments are integers. Returns a datetime with the date set to January 1, 1970). Can be in the form: ■ 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

Operator	Description	Usage example	Data type(s)
Keywords and functions			
			null if the coercion fails)
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	arg1 contains	string arguments, returns boolean

Operator	Description	Usage example	Data type(s)
Keywords and functions			
	pattern matching	arg2	
does not contain string	containment with pattern matching	arg1 does not contain arg2	string arguments, returns boolean
<code>==</code>	equality	<code>arg1 == arg2</code>	integer, long, decimal, string (case sensitive), or date
<code>=</code>	equality	<code>arg1 = arg2</code>	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)
<code>is</code>	equality	<code>arg1 is arg2</code>	integer, long, decimal, string (case sensitive), or date
<code>== null</code>	equality null test	<code>arg1 == null</code>	integer, long, decimal, string, or date
<code>= null</code>	equality null test	<code>arg1 = null</code>	integer, long, decimal, string, or date
<code>is null</code>	equality null test	<code>arg1 is null</code>	integer, long, decimal, string, or date
<code>!=</code>	inequality	<code>arg1 != arg2</code>	integer, long, decimal, string (case sensitive), or date
<code><></code>	inequality	<code>arg1 <> arg2</code>	integer, long, decimal, string (case sensitive), or date
<code>is not</code>	inequality	<code>arg1 is not arg2</code>	integer, long, decimal, string (case sensitive), or date
<code>!= null</code>	not null test	<code>arg1 != null</code>	integer, long, decimal, string, or date
<code><> null</code>	not null test	<code>arg1 <> null</code>	null integer, long, decimal, string, or date
<code>is not null</code>	not null test	<code>arg1 is not null</code>	integer, long, decimal, string, or date
Bitwise operators (<binary-op>) - processed left to right			
<code>&</code>	bitwise and	<code>arg1 & arg2</code>	integer or long
<code>^</code>	bitwise exclusive	<code>arg1 ^ arg2</code>	integer or long

Operator	Description	Usage example	Data type(s)
Keywords and functions			
	or		
	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.

Expression	Comment
time(hour, minute)	A time with the specified hour and minute (hours 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)

Method	Returns
getMinute()	integer, the minute within the hour (between 0 and 59)
getSecond()	integer, the second within the minute (between 0 and 59)
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
<code>compareToIgnoreCase(string)</code>	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

Method	Arguments	Type	Returns
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 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

Method	Arguments	Type	Returns
			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
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 of 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 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, perform some calculation across the entire array, and (usually) return a scalar value. For example:

```
int x = sum( item.lines[].properties.amount )
```

When performing 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

Operator	Description
	decimal values). The null aware version returns null if any value in the 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

Accessing information in external systems

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 the application, 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:
 - a. `com.cordys.entity.services.eisconnector.jar`
 - b. `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 AppWorks 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 SampleEISDefinitionProv  
();  
    /* 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 SampleEISDefinitionProv  
();  
    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 the application.

```
public EISEntityRow create(EISContext eisContext1, EISRepositoryEntity  
eisRespositoryEntity2, EISEntityRow eisEntityRow3)
```

Input parameters

- `eisContext1` - Contains details about the organization and the user who is trying to create the item.
- `eisRespositoryEntity2` - 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 AppWorks Platform organization name
- `getOrganizationId` – Returns the AppWorks Platform organization internal Id
- `getUserID` – Returns the AppWorks 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
eisRepositoryEntity2, 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 a result list in the application.

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 eisRespositoryEntity2, 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 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"
}
],
"criteria":{
    "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>
IsNotNull	Is Not Null	x is not null	<pre>"comparison": { "operator": "IsNotNull" }</pre>
InList	In (x1, x2, x3 ...)	x in (1, 2)	<pre>"comparison": { "operator": "InList", "values": [1, 2], "isValueExpression": false }</pre>
Between	Between x1 and x2	x between 1 and 10	<pre>"comparison": { "operator": "Between", "from": 1, "to": 10, "isFromExpression": false, "isToExpression": false }</pre>
ge	greaterthan or equal >=	x >= 1500	<pre>"comparison": { "operator": "ge", "value": 1500, "isValueExpression": false }</pre>
gt	greater than >	x > 1500	<pre>"comparison": { "operator": "gt", "value": 1500, "isValueExpression": false }</pre>

Comparator	Description	Sample SQL	
			<pre>"value": 1500, "isValueExpression": false }</pre>
eq	equals =	x = 'FULLTIME'	<pre>"comparison": { "operator": "eq", "value": "FULLTIME", "isValueExpression": false }</pre>
le	lessthan or equal <=	x <= 1500	<pre>"comparison": { "operator": "le", "value": 1500, "isValueExpression": false }</pre>
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
                }
              }
            ]
          }
        ]
      }
    ]
  }
]
```

```

"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":"IsNotNull"
      }
    },
    {
      "type":"PropertyTerm",
      "name":"Properties.x8",
      "comparison":{
        "operator":"NotInList",
        "values":[
          "Fred",
          "Ralph"
        ]
      }
    }
  ]
}
  ]
}
}

```

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 IN ('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 AppWorks Platform 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 *AppWorks Platform Advanced Development* 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 AppWorks 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 the application, 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](#)
- [Rule](#)

When a Rule building block is used, rules are triggered as follows:

Event	Description
A property changes	The rule is triggered when a form is first loaded and any time a

Event	Description
	property that is referenced in the rule is changed either in the application or in the external system.
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.
An instance is created	The rule is triggered when a new instance of the entity is created for that EIS entity from within the application. The rule does not fire when an instance for that EIS entity is created in the external system
An instance is deleted	The rule is triggered when an instance is deleted for that EIS entity from within the application. The rule does not fire when an instance for that EIS entity is deleted in the external system.

Relationships

To one and To many relationships can have an EIS entity as a target. An EIS entity itself cannot have outgoing relationships.

Identity for EIS entities

When an EIS entity is created, an Identity is added to it. All primary key properties of an EIS entity are created under the Identity of the EIS entity. The Identity also contains ItemId and EntityType properties.

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 *AppWorks Platform Advanced Development* documentation.

Maintaining the connection to the repository

Using [AppWorks Administration](#), you can associate an EIS repository reference with the appropriate EIS repository configuration after deployment.

To associate an EIS repository with the EIS repository configuration:

1. Open [AppWorks Administration](#).
2. Select an option from the **Show solutions deployed to** list, depending on the space where the package was deployed.
3. In the **Solutions** pane, select the solution 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**. 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.

AppWorks Administration

1. In [AppWorks Administration](#), 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 AppWorks Platform:

- Inbox
- Process Component Library

AppWorks 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 AppWorks 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.

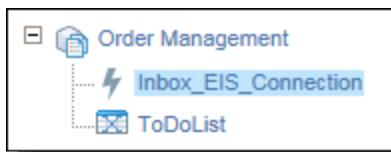
Note: Properties from a ToDoList entity created from the InBox EIS connection do not support searching or sorting.

To create a connection to an internal system entity:

1. In Workspace Documents, select your project.
2. Click **New**.
3. Select **Other**, search for Platform EIS Connection, and select it from the results.
4. Type the Display name, Name, and Description of the connection.

5. Click 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 **Add To do 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 **Add To do list**, a UserTask building block is added to the ToDoList entity.
If you click **Add Watch list**, 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

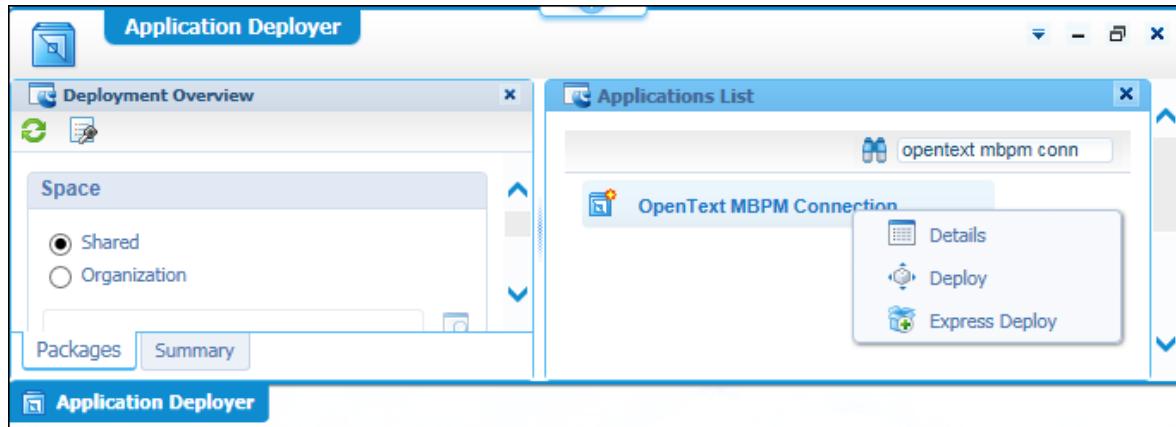
Use the MBPM Connection option to create entity types that model the information in the OpenText MBPM system. This enables users to retrieve and work with data in the MBPM system from the application.

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 AppWorks 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 Add To do list or Add 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 To do 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 Add 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.

To also remove the MBPM Connection and its entities from the application, you must delete the solution using [AppWorks Administration](#).

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.

Entity type	Description
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 MBPMAlert 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

Property	Data Type	Length
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 the application

Use [AppWorks Administration](#) to configure an MBPM connection so that it will be available within the application.

To configure an MBPM connection in the application:

1. Start [AppWorks Administration](#).
2. In the Solutions panel, select the solution that was created when the project was published or when the application package was deployed in AppWorks 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 AppWorks 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 AppWorks 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 AppWorks Platform OTDS Resource is selected (checked) in the OTDS Server.

For more detailed information, see the following topics in the *AppWorks Platform Advanced Development* 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 AppWorks Platform User. Additionally, UserName must be an MBPM MetastormAdministrator role.
- The AppWorks Platform username (the end user who signed in to AppWorks 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 the application to retrieve and work with data in the Case360 system.

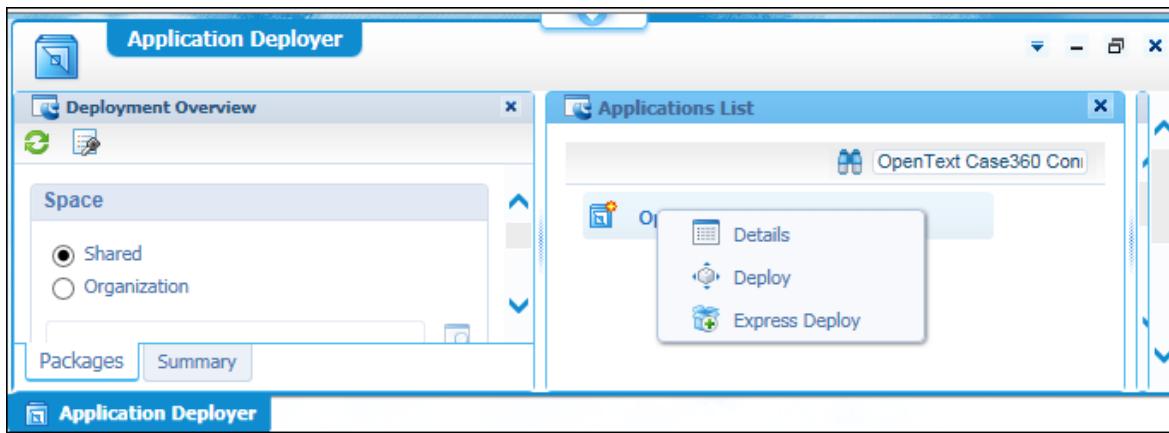
The Case360 connection supports Case360 version 11.3.0.38307 or greater.

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 AppWorks 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 AppWorks 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 the application.

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 the application:

1. Start [AppWorks Administration](#).
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 the application, you must delete the solution using [AppWorks Administration](#).

Property mapping

Case360 properties are mapped to the appropriate data type supported by the application as follows:

Case360 field type	Application data type
Boolean	Bool
Decimal	Decimal

Case360 field type	Application data type
Decimal Static Enumerated	Decimal
Decimal Dynamic Enumerated	Decimal
Duration	N/A
Integer Static Enumerated	Enumerated Integer
Integer Dynamic Enumerated	Enumerated Integer
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 13

Including advanced features

This section describes some advanced features that you can incorporate into your application development.

Designing iHub reports on entity data

OpenText AppWorks 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. AppWorks 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 [AppWorks Administration](#) and select the solution under **Solutions**.
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 Dimensions** 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, whether the solutions are deployed to the **My organization** or **Shared** spaces.

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.

Displaying reports in the application

You can display iHub reports in the application through the iHub panel.

To enable application users to view iHub reports:

1. Design a report and publish it to the iHub server.

See the OpenText Analytics documentation for details.

Note: If you are upgrading from Process Platform 16.3.1 to AppWorks Platform 16.4 or later versions, on publishing the BIRT project to the iHub server, provide full privileges to the Administrators user group on the Applications folder using the Share option in the iHub server. Without this, you will not be able to see the published report designs or dashboards in the iHub panel.

2. Configure the iHub Connection Manager to inform AppWorks Platform about the location of the iHub server so that it can connect to it.

See [Creating an organization](#).

3. Add a layout that contains an iHub panel to your project.

a. Open your project in Collaborative Workspace.

b. Add a layout.

c. Drag an iHub panel onto the layout and configure the general properties such as Name and Chrome.

d. 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.

e. Click  (Save).

4. Optionally, configure dynamic filtering for a report added to the layout. See [Configuring dynamic filtering](#) later in this topic.

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, [select the property](#) to be linked to the report parameter.
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.

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.

Publishing reports to other iHub environments

Reports are not maintained in the AppWorks Platform solution, nor are they packaged with the other artifacts in the AppWorks 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.

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 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 AppWorks Platform, 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 AppWorks Platform:

1. In AppWorks 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 the application.
3. Publish the project to the application.

4. Start the OCC Customizing process to perform the following configuration tasks:
 - Create a new entity.
 - Select AppWorks (previously called 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 the application.

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 email) as well as through hard copies.

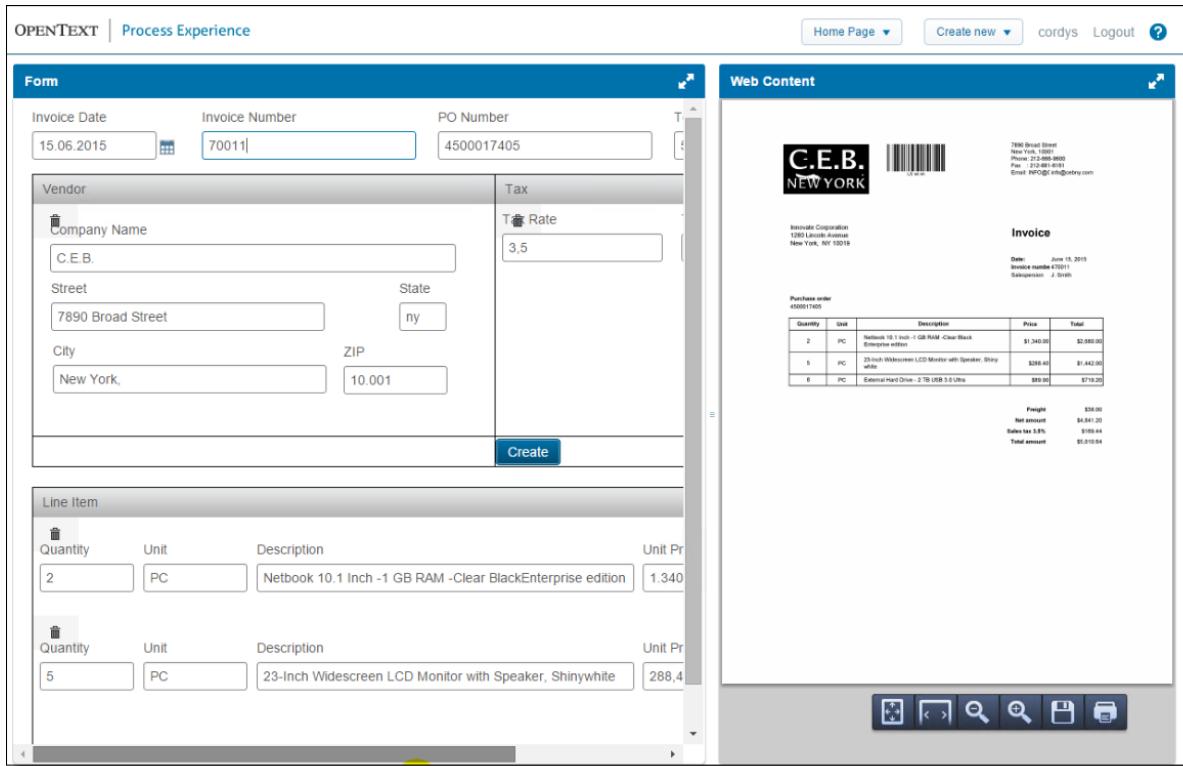
For the hard copies, you do not want to manually enter the details into the system. Instead, you let Capture Center scan each claim and create a corresponding Claim instance in the system. However, since the details on the hard copy are hand-written, there may be issues with digitizing the details. Therefore, 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 the application. You can control how the item is displayed using rules, forms, and layouts.

The following example shows an invoice that was created in the application using Capture Center, along with an image of the source paper-based document.



Configuring Extended ECM for AppWorks Platform

OpenText Extended ECM Platform helps build a foundation for an organization's digital transformation. It improves process productivity by integrating Enterprise Content Management (ECM) with lead applications to bridge silos and enhance information flows.

Users work in the business application they need, while simultaneously gaining access to all of the information they need to get their job done. Whether that information is stored on-premises or in a cloud application, the business gains a single source of the truth to help ensure compliance and control across all systems.

This section explains how to configure Extended ECM for AppWorks Platform.

Configuring a business workspace in AppWorks Platform

The basic steps to create and open a business workspace in AppWorks Platform are as follows:

1. Configure a business workspace in Content Server.
2. Create a document store application connector.
3. Open the business workspace in the application.

Configuring a business workspace in Content Server

To create a business workspace in Content Server from AppWorks Platform, configure Content Server as described in the *OpenText Extended ECM Platform Integration Guide*.

The high level configuration steps are as follows:

1. Connect an external system. See [Connecting an external system](#)
2. Create a category for the workspace type and the business object type. See [Creating a category](#)
3. Create a classification for document templates.
4. Create a location for the business workspaces.
5. Create a workspace type.
6. Define a document template for business workspaces.
7. Configure business object types.

Connecting an external system

To establish a connection from Content Server to AppWorks 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 AppWorks 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: <code>http://<server>:<port></code>
Application Server Endpoint	Specify the URL that will be called to obtain business object information. For example, for AppWorks Platform the URL is: <code>http://<server>:<port>/home/<organization>/app/xecm/ECMLinkService?WSDL</code>
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 AppWorks Platform entity properties and Content Server category attributes.

AppWorks 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 AppWorks Platform for mapping entity properties.

Creating a document store application connector

To create a document store application connector for Content Server, see *Creating and Modifying Document Store Repository* in the *AppWorks Platform Advanced Development* documentation.

If **Synchronize Workspaces Manually** is selected in the document store application connector configuration, an application 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 the application

To open a business workspace from the application, a user must be added to both AppWorks Platform and Content Server.

To open the business workspace in the application:

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 a 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 [AppWorks Administration](#). 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 AppWorks Platform](#).

To view a business workspace configuration report in AppWorks Platform:

1. Start [AppWorks Administration](#).
2. In **Solutions**, select the solution 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 AppWorks 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.
For Content Server 16.2 and greater, only business object types can be exported. For Content Server versions earlier than 16.2, you can export both business object types and workspace types.
 - c. Download the file.
2. Upload the Content Server Configuration file in AppWorks Platform.
 - a. Navigate to [AppWorks Administration](#) > 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 AppWorks Platform

To configure Content Server to work with business attachments in AppWorks 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 AppWorks Platform.

For example, [Properties.EntityProperty] creates business attachments with the EntityProperty name.

5. Click **Save Changes**.

Important: In Content Server, AppWorks Platform users should have permission to use documents as attachments in the application.

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 AppWorks Platform that needs to share or relate to a Content Server business workspace needs to configure the following settings in [AppWorks Administration](#).

The following sections explain how to configure a cross application business workspace.

To configure Content Server for cross application support in AppWorks 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 **Enable Queryable** for XECMWkspLinkRefTypeID.
 - Select **Displayable** for XECMWkspLinkSAPObjectKey.
3. Click **Update** to save the configuration.

To configure AppWorks Platform for cross application business support:

1. Start [AppWorks Administration](#).
2. In **Solutions**, select the solution 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 the application 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 AppWorks 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 an application user to link to Content Server business workspaces from entity items.

Note: The option to link an existing business workspace is available only if AppWorks 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 AppWorks Platform](#)

Configuring Content Server for Link business workspace support

To configure Content Server to link business workspace support in AppWorks 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 AppWorks Platform

This configuration enables an application user to link to Content Server business workspaces from entity items.

To configure a link to an existing business workspace:

1. Start [AppWorks Administration](#).
2. In **Solutions**, select the solution 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 [AppWorks Administration](#). If the same property mapping is done at both ends, data will be overwritten with the latest update.

Configuring metadata for Documentum

If you are using OpenText Documentum as the content repository, you can configure metadata and specify the properties of the metadata that must be displayed when the application user uses the Content panel for the Properties options. You can define conditions using a metadata property and specify the properties that must be displayed for the metadata when the conditions are met.

Note: You can configure the metadata only when an entity contains the File or Content building block, and the Document Store Application Connector is configured with Documentum as a repository.

To configure metadata in AppWorks Platform:

1. Start [AppWorks Administration](#).
2. In **Solutions**, select the solution that contains the entity.
3. In **Configurable elements > Entities**, select the entities for which to configure metadata.
The Metadata configurations section is displayed.
4. From **Specify a document type**, select the document type for which to configure properties.
All the properties of the selected document type are listed. In Select a condition, the value Default is already selected.

Note: Use the value Default when none of the defined conditions are applicable.

5. Click **Manage conditions**.
The Manage conditions dialog box opens.
6. Select a property from the **Property to set conditions on** drop-down list.
7. To add conditions, click **Add**.
8. Type the condition value and click the **Save** icon for it.
9. Save the changes to the Manage conditions dialog box.
10. From **Select a condition**, select the required condition.
11. Select the properties to be displayed at runtime.
The selected properties are auto-saved and displayed when the application user uses the Content panel for the Properties options.

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 the application.

For more information about each step, see the *AppWorks Platform Advanced Development* 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

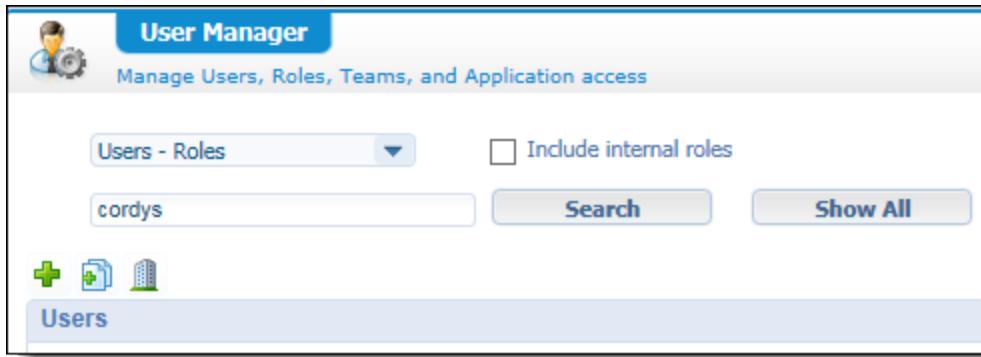
Creating an organization

An organization serves as a container where you can create and manage entities required to work with AppWorks 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.

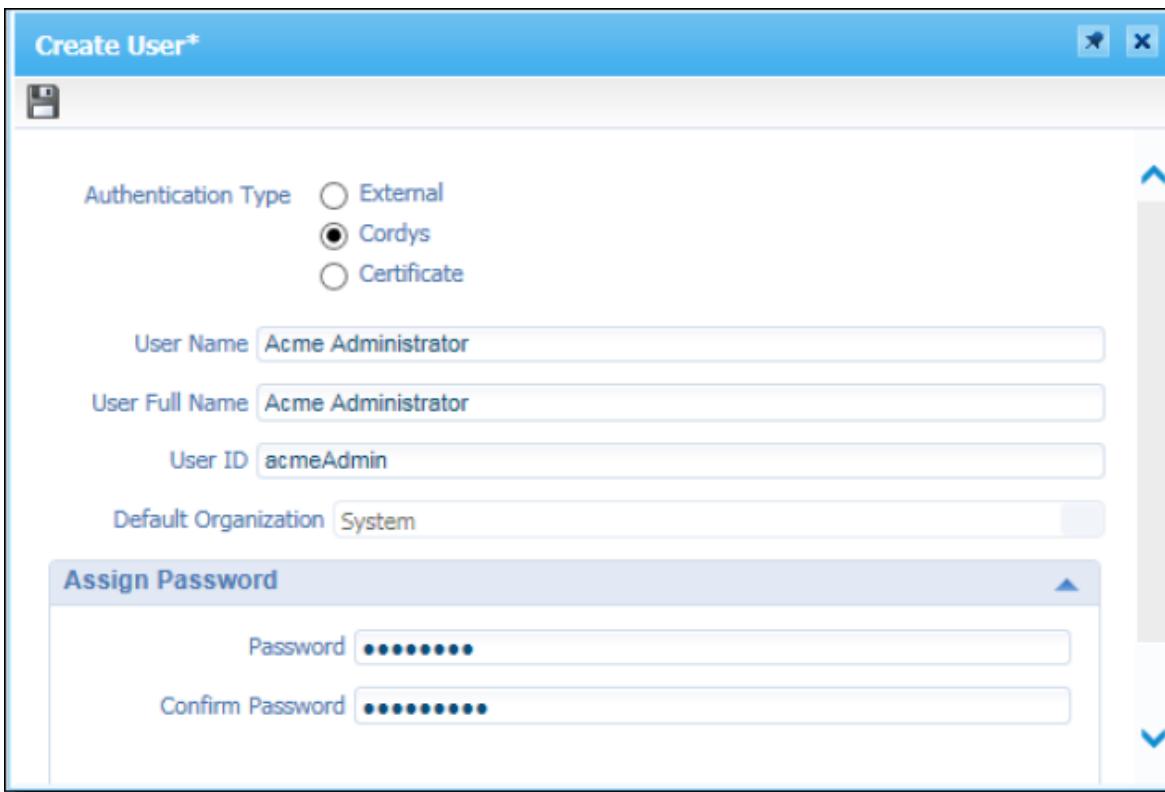
To assign the systemAdmin role:

1. In the system organization, start the AppWorks 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.

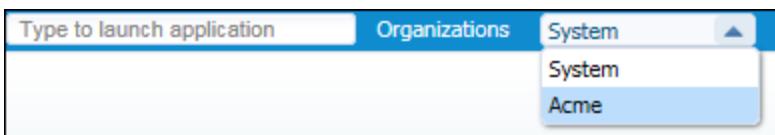


4. In Authentication Type, select **Cordys**.
5. Click  (Save).

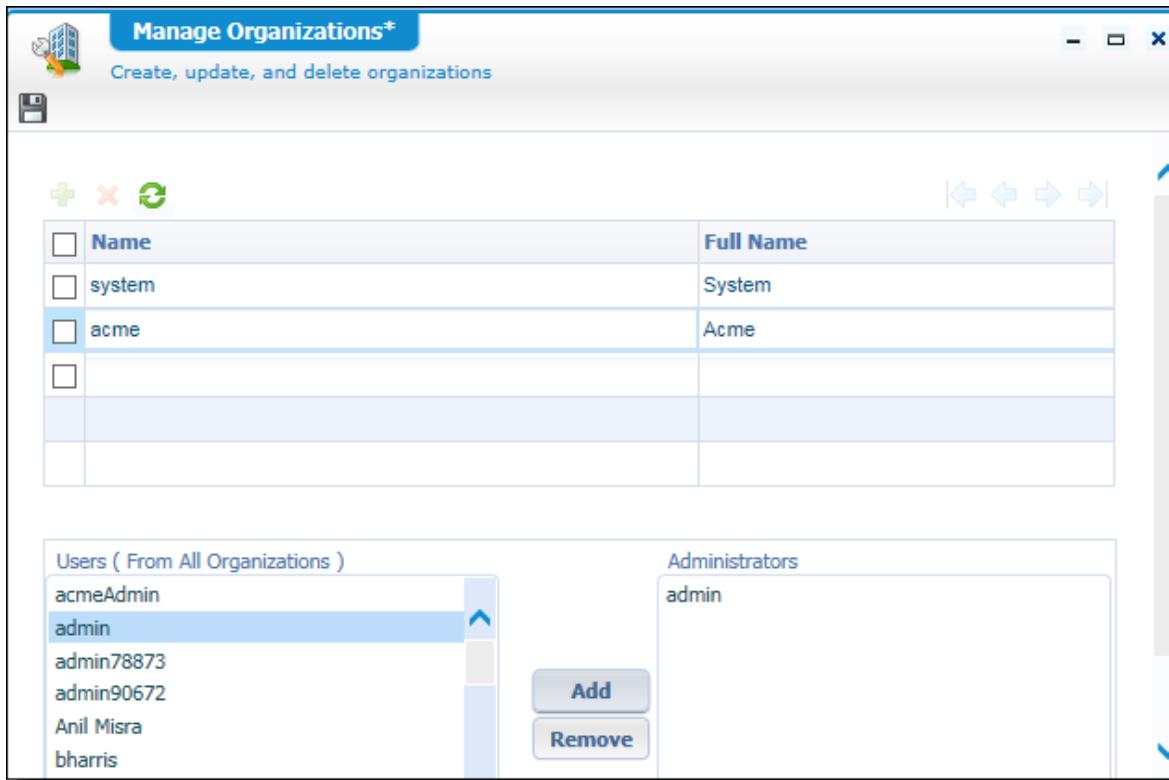
To create a target organization:

1. In the system organization, start the AppWorks 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 AppWorks Platform desktop instead of entering a different URL.



5. Click **Add**.



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 AppWorks 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 application 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 AppWorks 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 AppWorks 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 Protocol, Hostname, iPortal port number, REST port number, and Volume name of the iHub server you are connecting to.
The http Protocol option, iPortal port number, and REST port number are populated with default values. If iHub is configured on a different protocol and port, you must change the values accordingly.
5. Click **Save**.

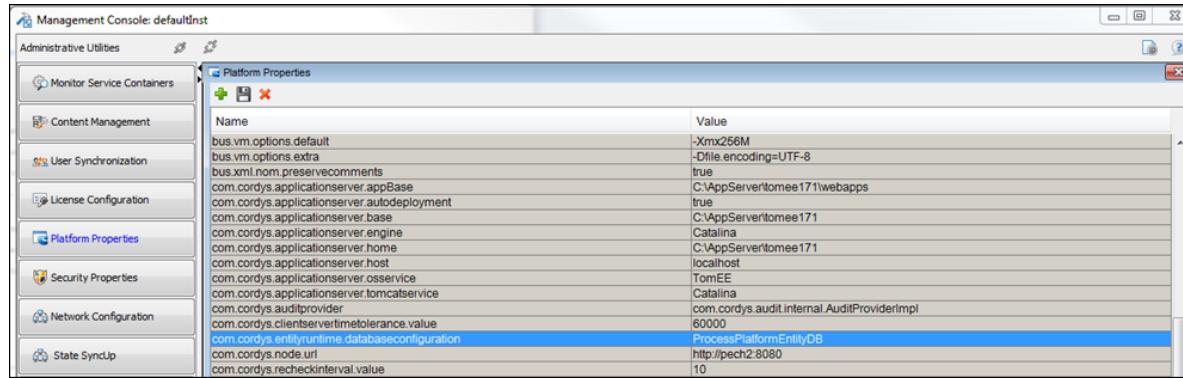
Configuring an entity runtime database per organization

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 AppWorks 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, `AppWorksPlatformEntityDB`).



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, AppWorksPlatformEntityDB).
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.

Name ProcessPlatformEntityDB	Description ProcessPlatformEntityDB
<input checked="" type="radio"/> JDBC <input type="radio"/> OLEDB	
JDBC Driver Microsoft SQL Server	Default Database acmedb
Driver Class com.microsoft.sqlserver.jdbc.SQLServerDriver	DB User sa
Connection String jdbc:sqlserver://pech2:1433	Password *****
JDBC Driver XA Class com.microsoft.sqlserver.jdbc.SQLServerXADataSource	

8. Click (Test Connectivity) to verify that AppWorks 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.

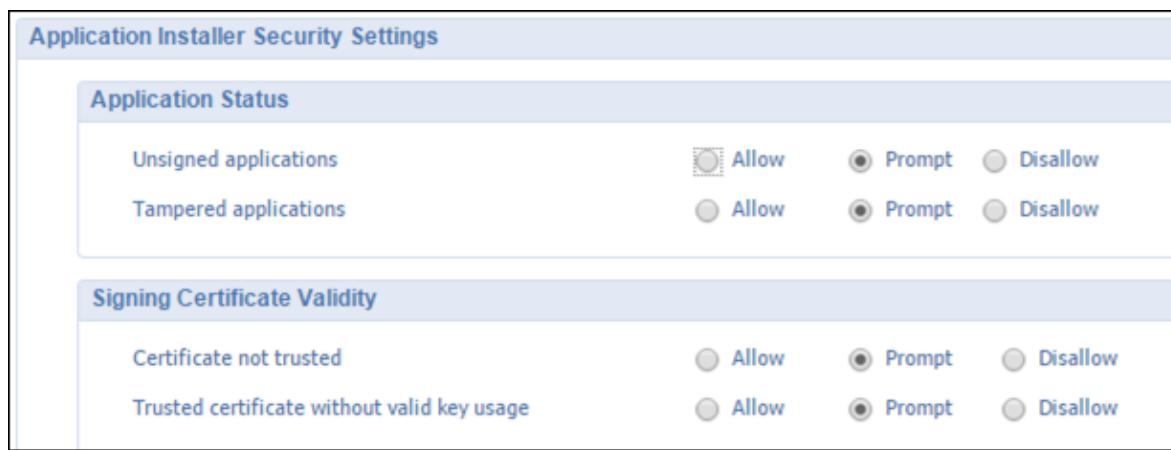
Deploying the application

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, in Organization, click  (Look up a value for), 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  (Lookup).
3. In the **Organizations List** dialog box, select the target organization and 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 [AppWorks Administration](#).
2. Click CTRL+F5 to clear the cache and refresh the display.
3. In the **Solutions** list in the left pane, select your solution.
4. In the **Configurable elements** list in the middle pane, expand **Solution security > 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 application user.

To test the application:

1. In the target organization, open the application.
2. Create a few items.

3. Open the application on the system organization and verify that the items you created are not shown.

Chapter 14

Maintaining the application

This section describes the tasks required to extend maintain your application.

Managing solutions

Use [AppWorks Administration](#) to perform many administrative functions for your applications. Applications are called solutions in AppWorks Administration. Solutions can be applications installed with the base product, applications deployed using Application Deployer, or projects published from AppWorks Platform Collaborative Workspace.

To run AppWorks Administration, type the following address in your browser's address box:

`http://<hostname>/home/<org>/app/admin`

To run AppWorks Administration, you need **Manage Solution Security** permission.

Use the AppWorks Platform User Manager to select the appropriate roles (select **Include internal roles** to see the additional roles).

To access AppWorks Administration so that you can set up the initial permissions and roles, you need the **Entity Runtime Administrator** role.

Defining security for a solution

When solutions (applications) are deployed or published, they are closed by default. To enable a solution in the application, assign the user to the **Entity Runtime User** role or to any other role and then use [AppWorks Administration](#) 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 [AppWorks Administration](#). 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

The application 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 Runtime Developer
Include	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

[AppWorks Administration](#) 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 AppWorks Administration tool by typing the following address in your browser's address box:
`https://<server>:<port>/home/<organization>/app/admin`
Replace the variables <shown in angle brackets> with the relevant values for your installation. For example: `https://localhost:8080/home/system/app/admin`
2. In the **Solutions** 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 AppWorks, a shared application called Inbox Task Management is automatically installed. The Inbox Task feature displays items from a user's AppWorks Platform Inbox in an application 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 the application, you need to set solution security for the Inbox Task Management application.

To set solution security for the Inbox Task Management application:

1. Open [AppWorks Administration](#).
2. From the **Show solutions deployed to** list, select Shared.
3. In **Solutions**, 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

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 AppWorks, a shared application called Identity package is automatically installed. This feature enables an administrator to manage users in the application.

Solution level security is managed out of the box during Identity package deployment. However, user level security, if needed, should be managed by the application administrator. See [Managing identities](#).

The following table describes the roles and corresponding features in the Identity package.

Name	Description
Identity Administrator	Subrole of Administrator. Enables managing user information from the Identity Homepage in the application.
Identity Push Service	Subrole of OTDS Push Service. Needed on OTDS Resource User configuration for synchronization.
Identity User	Enables access to the Inbox.

Defining solution variables

Any solution deployed in a shared space is shared by the entire organization. Organizations need to specify different values for the solution variables. Use [AppWorks Administration](#) to define solution variables that are subject to change, such as a discount rate. You can use solution variables in the following places:

- Email template building block
- Web content panel in the Layout building block
- URLs in action rules

The name of a solution variable must exactly match the name of the property that it represents. For example, if the solution variable represents a property called Discount, the solution variable must be named Discount.

To define solution variables:

1. Sign in to [AppWorks Administration](#).
2. In the **Solutions** panel, select one of the following options from **Show solutions deployed to**:
 - **My organization** (default option)
 - **Shared**

Note: If you select **Shared**, select an option from **Show configuration for**.

3. Select a solution from the listed solutions.
4. From **Configurable elements**, select **Solution variables**.
Each variable for that solution is listed in the Solution variables panel.
5. 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.

Editing a solution variable for tenants

By default, a tenant inherits the solution variables defined on a global level. As a tenant administrator, you can edit the values of the variables.

To edit a solution variable for tenants:

1. In a non-system organization, select a solution that is deployed in the shared space.
2. From **Configurable elements**, select **Solution variables**.
3. In the **Solution variables** panel, click **Replace**.
Note: The Replace option is only available for a solution selected from the shared space.
4. Click a row and edit the values.

Note: To reset the values to the global configuration, click **Revert**.

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 the application. 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 **Configurable elements**, select **Solution status > Errors 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 [AppWorks Administration](#), you can associate specific layouts in a solution with AppWorks Platform roles. This provides an additional layer of security to that provided by the Security building block.

To configure security for layouts:

1. In **Solutions**, select a solution.
The layouts and variables for that solution are listed in the Configurable elements panel.
2. 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, the layout becomes inaccessible.

Deleting a solution

Solutions listed in the [AppWorks Administration](#) Solutions 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: Solutions that were installed using the Application Deployer should only be removed using the Application Deployer undeploy feature.

You might need to delete a solution from the application 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 the application is deleted but your project remains intact in the CWS workspace and can be republished to the application.

Only users with **Build** permission can delete a workspace.

To delete a solution:

1. In the **Solutions** panel, select the solution to delete.
2. Click the gear icon and select **Delete**.
3. Click **OK** when prompted for confirmation.

Deleting an entity

You can delete an entity that was created by publishing a project from the CWS workspace during application development. For example, 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 solution.

Only users with **Build** permission can delete an entity.

To delete an entity:

1. In the **Solutions** 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

Sometimes a release will introduce new functionality in an entity building block that requires changes to the underlying data model. Such a change is not automatically incorporated into applications that were already deployed/published. The Administrator tool provides the Publish action to enable the administrator to apply the appropriate updates to the existing data model without creating and deploying a new version of the application.

To publish a solution:

1. In the **Solutions** panel, select the solution (application).
2. Click the gear icon and select **Publish**.

Deleting an EIS entity

You can delete an EIS entity created by publishing a project from the CWS workspace during application development. For example, 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 solution.

Only users with **Build** permission can delete an EIS entity.

To delete an EIS entity:

1. In the **Solutions** 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

You can delete a history log created by publishing a project in the CWS workspace during application development. For example, 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 solution.

Only users with **Build** permission can delete a history log.

To delete a history log:

1. In the **Solutions** 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 AppWorks Administration so that they can be applied to the database. When a solution has pending DDL statements, a new option called Pending database statements is displayed in the Solution status section.

The database administrator can take the statements shown and apply them to the database and, once they are applied, click the button. This notifies 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, use [AppWorks Administration](#) to populate the information for the email configuration, otherwise emails cannot be sent.

Thereafter, use AppWorks Administration to select an email configuration and provide appropriate values to the fields. For example, provide From, Display name, and other information.

To add or modify an email configuration:

1. In AppWorks Administration, select your solution from the **Solutions** list.
2. In **Configurable elements > Email configurations**, select an email configuration that you want to modify.
3. 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.

Note: To use the full name of the logged in user as the display name, select **Use the full name of the logged in user as the display name**.
 - c. In **Reply to**, type the email address to which replies will be sent.
 - d. In **Email profile user ID**, provide the profile user ID of the user, which can be obtained from the AppWorks Platform User Management. For example, jsmith.

This user credential 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.

Note: The SetProfile web service must be invoked from the same organization where the application is deployed. This step is mandatory if the email server requires authentication.

Editing email configuration for tenants

You can edit the inherited global configuration to create an organization-specific configuration.

To edit the inherited values:

1. In a non-system organization, select a solution that is deployed in the shared space.
2. In [AppWorks Administration](#), select your solution from the **Solutions** list.
3. In **Configurable elements > Email configurations**, select an email configuration that you want to modify.
4. Click **Replace**.
5. Click a row and edit the values.

Note: To reset the values to the global configuration, click Revert.

Managing identities

The Identity Package is an application that enables synchronization of user information (identities) from Open Text Directory Services (OTDS). It is automatically installed with AppWorks Platform and creates an entity for each type of user information. Users with the required privileges can use the Identity home page to maintain identity information.

You perform user management using the Identity Home Page layout in your application.

The Identity Package provides the following roles:

- The **Identity User** has Read permission on all entities in the Identity Package.
- The **Identity Administrator** has Read, Update, and Delete permissions for all the entities that are not synchronized from OTDS.

These roles are assigned to a User in AppWorks Platform User Manager.

Note: Any users that need to access the application must have the Entity Runtime User role. These roles also are assigned to a User in AppWorks Platform User Manager.

The Identity Home Page layout in your application provides access to the following lists:

▼ Identity

- All contacts
- All countries
- All email address types
- All emergency contact relationships
- All enterprises
- All genders
- All groups
- All languages
- All organizational units
- All phone number types
- All roles
- All social media
- All titles
- All users
- All visa types
- All worklists

The following identities are synchronized with OTDS. An Identity Administrator cannot create or edit these identities.

- User
- Role
- Group

User, Role, and Group information is directly synchronized with OTDS using the OTDS Push Connector for the OpenText Identity component. In addition, different relationships between User, Role, and Group are also saved to the User, Role, and Group entities that are created. For information about the OTDS Push Connector for OpenText CARS and Identity components, see the AppWorks Platform product documentation.

Only an OTDS Admin in OTDS can update users, roles, and groups and these will be synchronized to the Identity component. The OTDS Push Service Role is required to have a successful connection between OTDS and an AppWorks Platform instance.

User, role, and group information can be accessed from the Identity home page lists that are provided with the AppWorks Platform installation. Navigate to the Identity Homepage layout to access the default Identity lists.

The following identities are created when users are pushed. They can also be created from your application.

- Countries
- Genders

The following identities are not synchronized through OTDS. A user with the Identity Administrator role can create them from the Identity Home page.

- Enterprises
- Contacts
- Organizational units
- Worklists

The Identity Package also provides an identity called Address that can be associated with an organizational unit or a contact. The Address identity contains the following properties:

- City
- Note
- Postal Code
- Street Address

You create addresses when creating or editing an organizational unit, enterprise, or contact. Some additional features on the addresses created within organizational unit or contact follow:

- For an organizational unit with multiple addresses, you can specify a default address.
- If a contact has multiple addresses, you can specify one of them as the work address.
- For a user, the address is populated in OTDS. Therefore, only a single address is supported for users.

Contacts

A contact is a person who is a user of an external application rather than a user of AppWorks Platform. A contact does not use the application, but is part of the application data. For example, a contact cannot sign on to the application but may be used as an application data like shipment consignee. The contact information can be contextual based on the application (claimant, employee, and so forth).

The **All contacts** list shows the list of persons who are not system users.

To see a list of all contacts on the system:

- On the Identity Home page, open the **All contacts** list.

To select a contact:

- In the **All contacts** list, select the check box next to the contact name.
Basic information about the contact is shown in the Preview panel.

To open complete contact information:

- In the **All contacts** list, click the contact name link or select one or more contacts and click **Open**.
If you select multiple contacts, each one opens in a separate tab. The contact name, email address, and phone number are shown in the header area.

To create a contact:

Tip: When you create contacts, you will need to select a gender and country to include in the contact information. You can either create the genders and countries before you create the contact or you can add them later.

1. On the Identity Home page, select **Create > Contact**.
2. In the left pane, enter the contact information.
3. In All Addresses, create one or more addresses for the contact.
 - Click + (Create) and expand the sequentially numbered row that appears in the grid.
 - Enter the address information. If there is change in the country name for a previously selected Country, the State\Province also has to be updated for the updated country by selecting from the browse list. Otherwise, the State\ Province field will show the obsolete data.
 - If necessary, repeat these steps to create additional addresses.
4. In Work Address, click  (Browse) and select the work address.
5. Click **Create**.

Note: The State/Province can be added under the All State/Province list.

To edit contact information:

- Open the contact and make your changes.

To view the audit information of a contact:

- Open the contact and click **View History**.

To delete contacts:

- In the list of contacts, select one or more contacts and click **Delete**.
You can also delete a single contact by opening it and clicking **Delete**.

Countries

Countries are internally accessed by the User and Contact entities. A country can be created both as an internal property of a User through OTDS or as a standalone entity using the Identity Home page Create menu. The list of countries can be accessed using the standard Identity Home page lists.

To see a list of all countries on the system:

- On the Identity Home page, open the **All countries** list.

To select a country:

- In the **All countries** list, select the check box next to the country name.
Basic information about the contact is shown in the Preview panel.

To open complete country information:

- In the **All countries** list, click the country name link.

To create a country:

1. On the Identity Home page, select **Create > Country**.
2. In Country name, type the name of the country.
3. In ISO country code, type the ISO code for the country (the value cannot exceed two characters).
4. Click **Create**.

To add a state or province for a country:

1. Open the country.
2. In All State/Province, click + (Create).
3. In the new row that appears, type the name of the state or province.

To edit country information:

- Open the country and make your changes.

To view the audit information of a country:

- Open the country and click **View History**.

To delete countries:

- In the list of contacts, select one or more contacts and click **Delete**.
You can also delete a single contact by opening it and clicking **Delete**.

Email address type

The email address type is used when creating an email address. Entries might be Personal, Official, and so forth.

To see a list of all email address types on the system:

On the Identity Home page, open the **All email address types** list.

To select an email address type:

- In the **All email address types** list, select the check box next to the email address type.

To open an email address type:

- In the **All email address types** list, click the email address type link or select one or more email address types and click **Open**.

If you select multiple items, each one opens in a separate tab.

To create an email address type:

1. On the Identity Home page, select **Create > Email address type**.
2. Type the name of the email address type.
3. Click **Create**.

To edit email address type information:

- Open the email address type and make your changes.

To delete email address types:

- In the list of email address types, select one or more items and click **Delete**. You can also delete a single item by opening it and clicking **Delete**.

Emergency contact relationship

The **All emergency contacts** list shows all the emergency contact relationships on the system.

To see a list of all emergency contact relationships on the system:

- On the Identity Home page, open the **All emergency contact relationships** list.

To select an emergency contact relationship:

- In the **All emergency contact relationships** list, select the check box next to the emergency contact relationship name.

To open emergency contact relationship information:

- In the **All emergency contact relationships** list, click the emergency contact relationship link or select one or more emergency contact relationships and click **Open**. If you select multiple items, each one opens in a separate tab.

To create an emergency contact relationship:

1. On the Identity Home page, select **Create > emergency contact relationship**.
2. Type the name of the emergency contact relationship.
3. Click **Create**.

To edit emergency contact relationship information:

- Open the emergency contact relationship and make your changes.

To delete emergency contact relationships:

- In the list of emergency contact relationships, select one or more items and click **Delete**. You can also delete a single item by opening it and clicking **Delete**.

Enterprises

In the Identity Package, an enterprise is a type of organizational unit. A single tenant contains one or more enterprises. One enterprise is the owner of the tenant. An enterprise is sometimes called a company or an organization and it may consist of other organizations such as divisions. An Enterprise is the root for the organizational units.

The All enterprises list shows all the enterprise lists on the system.

To view the list of enterprises:

- On the Identity home page, open the **All enterprises** list. When you select an enterprise in the Results panel, basic information is displayed in the Preview panel.

To open complete enterprise information:

- In the All enterprises list, click the enterprise name link.

To create an enterprise:

1. On the Identity home page, click **Create > Enterprise**.
2. Type a name and description for the enterprise.
3. In Type, select **Hierarchical** or **None**.

4. In All Addresses, create one or more addresses for the enterprise.
 - Click + (Create) and expand the sequentially numbered row that appears in the grid.
 - Enter the address information. If there is change in the country name for a previously selected Country, the State\Province also has to be updated for the updated country by selecting from the browse list. Otherwise, the State\ Province field will show the obsolete data.
 - If necessary, repeat these steps to create additional addresses.
5. In Default Address, click  (Browse) and select the default address.
6. Click **Create**.

To delete an enterprise:

- In the list of enterprises, select one or more enterprises and click **Delete**. You can also delete a single enterprise by opening it and clicking **Delete**.

To view audit information and history of an enterprise:

- Click **History**.

Detailed information about each enterprise is shown on the following tabs.

Summary tab

The Summary tab displays summary information about the selected enterprise. You can update details such as name, description, address, and so forth. You can also add a new address.

Positions tab

To create various positions that belong to the current enterprise, navigate to the Positions tab. For example, the position can be CEO, CTO, Managing Director, President, and so forth. Additionally, you can explicitly set a position (such as Managing Director) as a lead position in the current enterprise.

To view a list of positions:

1. Open the Positions tab.
The list of positions is shown at the bottom of the window. You can filter the list by clicking the Actions menu and selecting or clearing columns from the list that is displayed.
2. If you want to view an audit trail for a position, select it and click **View History**.
3. If you want to delete a position, select it and click **Delete**. You cannot delete a position if it is in use.

To create a position:

1. In the list of positions, click + (Create).
2. In the blank row that appears, type the Name and Description of the position and select true or false to specify whether this is a lead position.

Members tab

To assign users to various positions, navigate to the Members tab and browse for the users.

To view a list of member assignments:

- Open the Members tab.
The list of positions and the members assigned to them is shown at the bottom of the window.

To assign a user to a position:

1. On the Members tab, select a position.
2. In the Assignments area, click + (Create).
3. Expand a row, click  (Browse) and select a user from the list that appears.
4. Click **Select**.
5. Select a Start Date and End Date and choose true or false to
6. In the blank row that appears, type the Name and Description of the position and select true or false to specify whether this is the primary organization for the user.
7. If you want to view an audit trail of an assignment, select it and click **View History**.
8. If you want to delete an assignment, select it and click **Delete**.

Sub Units tab

Use the Sub Units tab to view, associate, or remove an organizational unit as a sub unit of the current enterprise. For example, For example, an enterprise might have different regions based on organizational units in different locations.

Genders

Genders are internally accessed by the User and Contact entities. A gender can be created both as an internal property of a User through OTDS or as a standalone entity using the Create menu on the Identity Home page. The list of genders can be accessed using the standard Identity Home page lists.

To see a list of all genders on the system:

- On the Identity Home page, open the **All genders** list.

To select a gender:

- In the **All genders** list, select the check box next to the gender name.
Basic information about the gender is shown in the Preview panel.

To open complete gender information:

- In the **All genders** list, click the gender name link or select one or more genders and click **Open**.

To create a gender:

1. On the Identity Home page, select **Create > Gender**.
2. Type the name of the gender.
3. Click **Create**.

To edit gender information:

- Open the gender and make your changes.

To view the audit information of a gender:

- Open the gender and click **View History**.

To delete genders:

- In the list of genders, select one or more genders and click **Delete**. You can also delete a single gender by opening it and clicking **Delete**.

Groups

The **All groups** list shows a list of all the groups that are available in OTDS. When you select a group, the Preview panel displays the name and description of the current group.

To view more details about the group, open the group. Additional details are displayed on the following tabs.

Members tab

The Members tab displays the users, roles, and groups associated with the current group.

Member of tab

The Member of tab shows any other groups of which the current group is a member.

Language

The **All Languages** list shows all the languages that are available on the system.

To see a list of all languages on the system:

- On the Identity Home page, open the **All languages** list.

To select a language:

- In the **All languages** list, select the check box next to the name of the language.

To open language information:

- In the **All languages** list, click the language link or select one or more languages and click **Open**.

If you select multiple languages, each one opens in a separate tab.

To create a language:

1. On the Identity Home page, select **Create > Language**.
2. Type the name and ISO language code of the language.
3. Click **Create**.

To edit language information:

- Open the language and make your changes.

To delete languages:

- In the list of languages, select one or more items and click **Delete**. You can also delete a single item by opening it and clicking **Delete**.

Organizational units

An organizational unit is a group of users who form a functional team. Organizational units have qualifying designations called Positions that can in turn be assigned to Persons.

To view the list of organizational units:

- On the Identity home page, open the **All organizational units** list. When you select an organizational unit in the Results panel, basic information is displayed in the Preview panel.

To create an organizational unit:

1. On the Identity home page, click **Create > Organizational unit**.
 2. Type a name and description for the organizational unit.
- Note:** The name cannot contain the character '/'.
3. In Type, select **Hierarchical** or **None**.
 4. In All Addresses, create one or more addresses for the organizational unit.
 - Click + (Create) and expand the sequentially numbered row that appears in the grid.
 - Enter the address information. If there is change in the country name for a previously selected Country, the State\Province also has to be updated for the updated country by selecting from the browse list. Otherwise, the State\ Province field will show the obsolete data.
 - If necessary, repeat these steps to create additional addresses.

5. In Default Address, click  (Browse) and select the default address.
6. Click **Create**.

To delete organizational units:

- In the list of organizational units, select one or more organizational units and click **Delete**.
You can also delete a single organizational unit by opening it and clicking **Delete**.

To view audit information and history of an organizational unit:

- Click **History**.

Detailed information about each organizational unit is shown on the following tabs.

Summary tab

The Summary tab displays summary information about the selected organizational unit. You can update details such as name, description, address, and so forth. You can also add a new address.

Positions tab

To create various positions that belong to the current organizational unit, navigate to the Positions tab. For example, the position can be Manager, Engineer, Tester, and so forth. Additionally, you can explicitly set a position (such as Manager) as a lead position in the current organizational unit.

To view a list of positions:

1. Open the Positions tab.
The list of positions is shown at the bottom of the window. You can filter the list by clicking the Actions menu and selecting or clearing columns from the list that is displayed.
2. If you want to view an audit trail for a position, select it and click **View History**.
3. If you want to delete a position, select it and click **Delete**. You cannot delete a position if it is in use.

To create a position:

1. In the list of positions, click + (Create).
2. In the blank row that appears, type the Name and Description of the position and select true or false to specify whether this is a lead position.

Members tab

To assign users to various positions, navigate to the Members tab and browse for the users. If a user is part of more than one organizational unit, one of the organizational units can be set as the principal organizational unit to which the user primarily belongs.

To view a list of member assignments:

- Open the Members tab.
The list of positions and the members assigned to them is shown at the bottom of the window.

To assign a user to a position:

1. On the Members tab, select a position.
2. In the Assignments area, click + (Create).
3. Expand a row, click  (Browse) and select a user from the list that appears.
4. Click **Select**.
5. Select a Start Date and End Date and choose true or false to
6. In the blank row that appears, type the Name and Description of the position and select true or false to specify whether this is the primary organization for the user.
7. If you want to view an audit trail of an assignment, select it and click **View History**.
8. If you want to delete an assignment, select it and click **Delete**.

Sub Units tab

Use the Sub Units tab to view, associate, or remove an organizational unit as a sub unit of the current organizational unit. For example, a Human Resources organization might have sub units called Benefits, Education, Onboarding, and so forth.

Phone number type

The phone number type is used to select a type when creating a phone number. Entries might be Home, Work, Mobile, and so forth.

To see a list of all phone number types on the system:

- On the Identity Home page, open the **All phone number types** list.

To select a phone number type:

- In the **All phone number types** list, select the check box next to the phone number type name.

To open complete phone number type information:

- In the **All contacts** list, click the phone number type link or select one or more phone number types and click **Open**.

If you select multiple phone number types, each one opens in a separate tab.

To create a phone number type:

1. On the Identity Home page, select **Create > Phone number type**.
2. Enter a phone number type and click **Create**.

To edit a phone number type:

- Open the phone number type and make your changes.

To delete phone number types:

- In the list of phone number types, select one or more items and click **Delete**. You can also delete a single item by opening it and clicking **Delete**.

Roles

The **All roles** list displays a list of all the roles that are synchronized with OTDS using the OTDS Push Roles feature in the Security Administration task. For more information, see the Synchronizing roles from AppWorks Platform to OTDS topic in the *AppWorks Platform Administration Guide*. On selecting a specific role, the Preview panel displays the following basic information about the selected role.

- Role name
- Description

To view additional details about the user, click the role name. A new window opens that displays additional details about the role on the following tabs.

Summary tab

The Summary tab displays the following information:

- Role name
- Description

Members tab

The Members tab displays the users and groups that are associated with the role.

Social media

Social media is used to select a type of social media. Entries might be LinkedIn, Facebook, Twitter, and so forth.

To see a list of all social media on the system:

- On the Identity Home page, open the **All social media** list.

To select a social media:

- In the **All social media** list, select the check box next to the social media name.

To open social media:

- In the **All social media** list, click the social media link or select one or more social media and click **Open**.

If you select multiple social media, each one opens in a separate tab.

To create social media:

1. On the Identity Home page, select **Create > Social media**.
2. Enter the name of the social media and click **Create**.

To edit social media:

- Open the social media and make your changes.

To delete social media:

- In the list of social media, select one or more items and click **Delete**.
You can also delete a single item by opening it and then clicking **Delete**.

Title

The **All titles** list displays information about the titles that are available on the system. Entries might be B.S., M.D., Ph.D., D.D.S, and so forth.

To see a list of all titles on the system:

- On the Identity Home page, open the **All titles** list.

To select a title:

- In the **All titles** list, select the check box next to the title.

To open complete title information:

- In the **All titles** list, click the title link or select one or more titles and click **Open**.
If you select multiple titles, each one opens in a separate tab.

To create a title:

1. On the Identity Home page, select **Create > Title**.
2. Enter the title and click **Create**.

To edit a title:

- Open the title and make your changes.

To delete titles:

- In the list of titles, select one or more items and click **Delete**.
You can also delete a single item by opening it and clicking **Delete**.

Users

The **All users** list provides information about all the users on the system. When you select a user (click the check box next to the user name), a Preview panel displays the following

information about that user. If you multiple users are selected, each user is shown on a separate tab.

- User name
- Gender
- Birthdate
- Email
- Phone

To view additional details about the user, click the user name. A new window opens that displays additional details about the user on the following tabs.

Summary tab

The Summary tab displays the following information:

- First name
- Last name
- Display name
- Gender
- Birthdate
- Email
- Phone
- All Addresses
 - Street Address
 - City
 - Postal Code
 - Country
 - State/Province
- Work Address

Roles tab

The Roles tab displays the roles that are assigned to the user.

Groups tab

The Groups tab displays the groups that are associated with the user

Visa types

The **All visa types** list displays a list of all the visa types available on the system.

To see a list of all visa types on the system:

- On the Identity Home page, open the **All visa types** list.

To select a visa type:

- In the **All visa types** list, select the check box next to the social media name.

To open visa types:

- In the **All visa types** list, click the visa type link or select one or more visa types and click **Open**.

If you select multiple visa types, each one opens in a separate tab.

To create visa types:

1. On the Identity Home page, select **Create > Visa type**.
2. Enter the name of the visa type and click **Create**.

To edit a visa type:

- Open the visa types and make your changes.

To delete visa types:

- In the list of visa types, select one or more items and click **Delete**.
You can also delete a single item by opening it and then clicking **Delete**.

Worklists

A WorkList is a list of workitems that will be serviced by assigned organizational units. The **All worklists** list shows all the worklists on the system.

To see a list of all worklists on the system:

- On the Identity Home page, open the **All worklists** list.

To select a worklist:

- In the **All worklists** list, select the check box next to the worklist name.
Basic information about the worklist is shown in the Preview panel.

To open complete worklist information:

- In the **All worklists** list, click the worklist name link.

To create a worklist:

1. On the Identity Home page, select **Create > Worklist**.
2. In Worklist name, type the name of the worklist.
3. In Description, type a description of the worklist.
4. Click **Create**.

To view an audit trail of a worklist:

- Open a worklist and click **View History**.

To delete a worklist:

- Remove the associated organizational unit and click **Delete**.

To view the complete details of the worklist, and to associate the organizational units or a worklist manager, open the worklist item. The worklist information is shown on the following tabs.

Summary tab

The Summary tab displays a summary of worklist details.

Units tab

The Units tab shows the organizational units that are associated with the current worklist item.

To associate an organizational unit with the current worklist item:

- On the Units tab, click  (Browse) and choose one or more organizational units.

Lead Positions tab

You can set one or more positions of the organizational unit as a lead of the worklist or a worklist manager.

To set a lead position:

1. On the Lead Position(s) tab, click  (Browse) and select a position from the list that appears.
2. Click **Select**.

Chapter 15

Tips and tricks

This section provides guidelines and instructions for performing specific functions in your entity-based applications.

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 the application. Multiple entities can share a single master data list. For example, an entity called Jobs can contain a master list of job titles. This frees users from having to enter the job title directly each time they create an item. It also ensures that the job title is entered consistently, regardless of the type of item a user is creating.

Before you begin:

If you do not already have an entity that you can use to create an item in the application, 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 Job Title as an example of master data and Employee as an example of an application that access the master data.

To create a Job Title master data entity:

1. Create an entity called Job Title. See [Creating entities](#).
 - Add text properties called Job Title (do not change the default length) and Job Code (set length to 4). See [Adding properties](#).
 - Add a list called All Job Titles and include the Job Title and Job Code properties. See [List](#). In an actual production environment, you might also want to include other properties such as Job Description.
 - Add a Create form that includes the Job Title and Job Code properties. See [Adding forms](#).
2. Open the Employee entity.
 - Add a To one relationship called Job Title and select JobTitle as the related entity. See [Adding relationships](#).
 - Edit the Employee entity's Create form to include the Job Title property and folder. Select **Show Browse** for the icon and **All Job Titles** for the list. The drop list

presentation on a To one Relationship 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 the application:

1. Open the application and refresh the display.
2. Select **Create new > Job Title** and create a few job titles.
3. Create a new employee and click  (Browse) to select a job title.
4. Open the employee item that you created and notice that the job title is displayed. You can change the job title by clicking  (Browse).
5. Open the All Job titles list. You see a list of all the job titles 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 related entity to use to get a list of items in the application. 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 the application, 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 application 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 email 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 email to Assignee notifying about the assignment.
3. If PreviousAssignee != null, send email to PreviousAssignee notifying that an item is unassigned from PreviousAssignee.
4. Update PreviousAssignee = Assignee.

Validating email address properties

It is not uncommon to use a text property to hold email addresses. When you do so, remember to include validation to ensure users enter valid email 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
```


Chapter 16

Changing the data type of a primary key column of an imported database table

AppWorks Platform supports the import of database tables as entities (called 'imported entities'). The ID of an imported entity is the primary key column(s) of the database table. An imported entity can be extended with forms, rules, and so forth and be referenced by other entities. If the data type of a primary column of a database table needs to be changed, the imported entity created from it must be updated. This update can impact the modeled existing content.

Impact of changing the datatype of a primary key column

The primary key is used in relationships from native entities to the imported entity. If you have this type of relationships, you need to delete the solution from the entity runtime when the data type of a primary key has changed. You must also delete any other solutions that depend on the deleted solution.

Important: When a solution is deleted, all entity tables and their contents are deleted from the database. Therefore, the data type of a primary key should be changed only during development of an application. When you do this for an application that is in production, all production data is lost.

To delete a solution from the entity runtime:

1. Open [AppWorks Administration](#).
2. In the **Solutions** pane, select the solution.
3. Click  (Tools) and then click **Delete**.

Enabling updating of the data type of a primary key of an imported entity

By default, an imported entity cannot be updated if the data type of the primary key has been changed. To enable updating, the following parameter must be set in the AppWorks Platform properties file:

```
feature.toggle.entity.importer.primarykey.type.change.supported=true
```

To set the parameter in the AppWorks Platform properties file:

1. Open the Management Console.
2. Select **Platform Properties**.
3. Click  (Add) in the form that opens and enter the parameter and value.

```
feature.toggle.entity.importer.primarykey.type.change.supported=true
```

4. Click **OK**.

To update the project in CWS:

1. From the context menu on the document, select **Reload Database Metadata** to reload the database schema into your Database metadata document.
2. Open the entity in the entity modeler and click **Update**.
3. Clean the build output.
4. [Delete the existing solution](#) as described above.
5. Publish the project.

Chapter 17

Data structure

One positive aspect of entity modeling is that it removes the need to work with databases and tables directly. It provides a functional layer that is easy to use and hides the technical complexity of databases, tables, columns, references, and indexes.

As the designer of entity-based applications, you are not required to know about what happens behind the scenes. However, this information can be useful and may impact certain design decisions. For example, it could be important to know how the entity name is related to the database table to prevent problems later.

When you install AppWorks Platform, entity modeling uses its own database that is separate from AppWorks Platform. It is important to understand what gets created in the database so that you can keep this in mind when creating a common entity naming convention in your design.

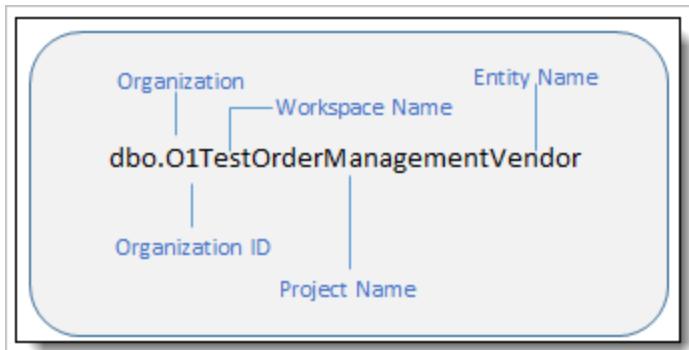
The following table shows the tables that are used within entity modeling.

Table prefix	Description
BAM	System tables.
O	Created for entities.
S_	System tables needed by the entity run time.
SHRD	Created when you publish to a shared space. Used by all tenants.
SQ	One is created for each item to keep track of the organization IDs. There is one SQ table for every O table.

Tables

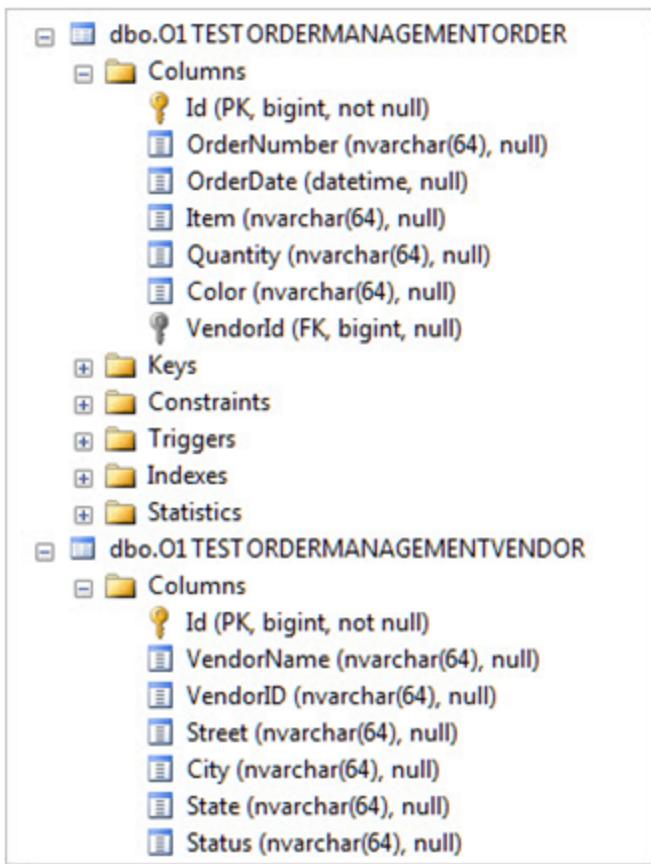
Tables are automatically created in the database when you create an entity-based application. Even though you should never modify the database tables directly, it is important to understand how the database is structured.

When you create entity-based applications, the following naming convention is used. Keep in mind that the dbo prefix in the following example is SQL Server specific.



As you add to your entity, columns are added to the table.

The following figure shows the database tables created for a simple Order Management application with two entities: Order and Vendor.



Columns are automatically added to the database table when you add certain building blocks to your entities. Some examples include:

- When you add a Title building block to an entity, the S_TITLE column is added to the table.
- When you add the File building block to an entity, the following columns are automatically added: s_FILENAME, s_FILENAME, and s_STORAGE.

When an entity is published, the properties defined for it are written as a column along with the properties exposed by the building blocks.

Note: Oracle and SQL Server have different requirements for table names. Since Oracle requires shorter table names, the names are automatically truncated to meet these requirements.