Configuring Solutions

STUDENT GUIDE

aras INNOVATOR

Copyright © 2024 by Aras Corporation. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at http://www.opencontent.org/openpub/).

Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.

Distribution of the work or derivative of the work in any standard (paper) book form for a commercial purpose is prohibited unless prior permission is obtained from the copyright holder.

Aras Innovator®, Aras, and the Aras Corporation "A" logo are registered trademarks of Aras Corporation in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners.

Microsoft, Office, SQL Server, IIS, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Notice of Liability

The information contained in this document is distributed on an "As Is" basis, without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement. Aras shall have no liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this document or by the software or hardware products described herein.

Revision October 2024

Table of Contents

Unit 1 Introduction 5
Unit 2 Users and Identities 13
Unit 3 Exploring ItemTypes 27
Unit 4 The Table of Contents (TOC) Editor 45
Unit 5 Itemtype Properties 57
Unit 6 Creating Object Properties 79
Unit 7 Creating the User Interface 93
Unit 8 Creating Item Relationships 123
Unit 9 Assigning Security 143
Unit 10 Visualizing Data 165
Unit 11 Dashboards and Reporting 193
Unit 12 Defining Extended Classification 221
Unit 13 Defining Lifecycle Maps and Versioning 245
Unit 14 Creating a Workflow Map 271
Unit 15 Enhancing a Workflow 301
Unit 16 Working with Files 319
Unit 17 Replicating Files 337
Unit 18 Sending Notifications 361
Unit 19 Implementing Poly Items 379
Unit 20 Configuring Command Bars 393
Unit 21 Packaging a Solution 415
List of Exercises and Labs 439
Appendix A Review and Lab Questions Answer Key 447

This page intentionally left blank

Unit 1 Introduction

Introduction

In this five-day course, you will learn the fundamental concepts and skills needed to configure, manage, and extend the Aras Innovator environment. Through hands-on exercises and lab activities, you gain an understanding of how Aras Innovator can be used to create a variety of business solutions.

Course Goals

- Understand and create an ItemType.
- Create a user interface that allows operators to interact with the solution.
- Assign appropriate security policies to a solution.
- Define Classifications, Lifecycles, and versioning.
- Create and enhance Workflow maps.
- Support file upload/download as well as file replication.
- Automatically generate email correspondence.
- Define and implement Poly Items.
- Configure Command Bars.
- Design, develop and package a solution that incorporates a Lifecycle, supports versioning, and uses a workflow.

The Configuring Solutions Course teaches you how to design and create simple solutions built with custom categories (ItemTypes) that include forms, properties and relationships to other items being managed by Aras Innovator. You will also learn how to replicate business processes by creating workflows and to create a custom security model to define who can access or edit data. To make sure that other Aras Innovator users get informed when they have tasks to perform, you will learn to set up automated email notifications. We will also discuss how Aras Innovator can be configured to manage the secure storage of third-party files in one or more vaults. The process of creating file replication and distribution rules will be demonstrated in class exercises.

Friendly Reminders

- Aras Innovator runs on Chrome, Firefox and Edge. Use recent browser versions
- Ensure you are logged into the database assigned to you
- Class consists of:
  - 2 daily instructor-led sessions
  - Labs to be completed outside instructor-led times
- Ask questions
- Feedback is encouraged

aras

6 | © 2023 Aras

Aras Innovator Requirements

The Aras Innovator client is browser agnostic: it runs in Microsoft Edge, Mozilla Firefox, or Google Chrome browsers.

Please refer to the *Aras Innovator Installation Guide* and Client Settings documents for additional technical information and system requirements. It is recommended to use the latest version of Chrome and Edge, but only ESR versions of Firefox are supported.

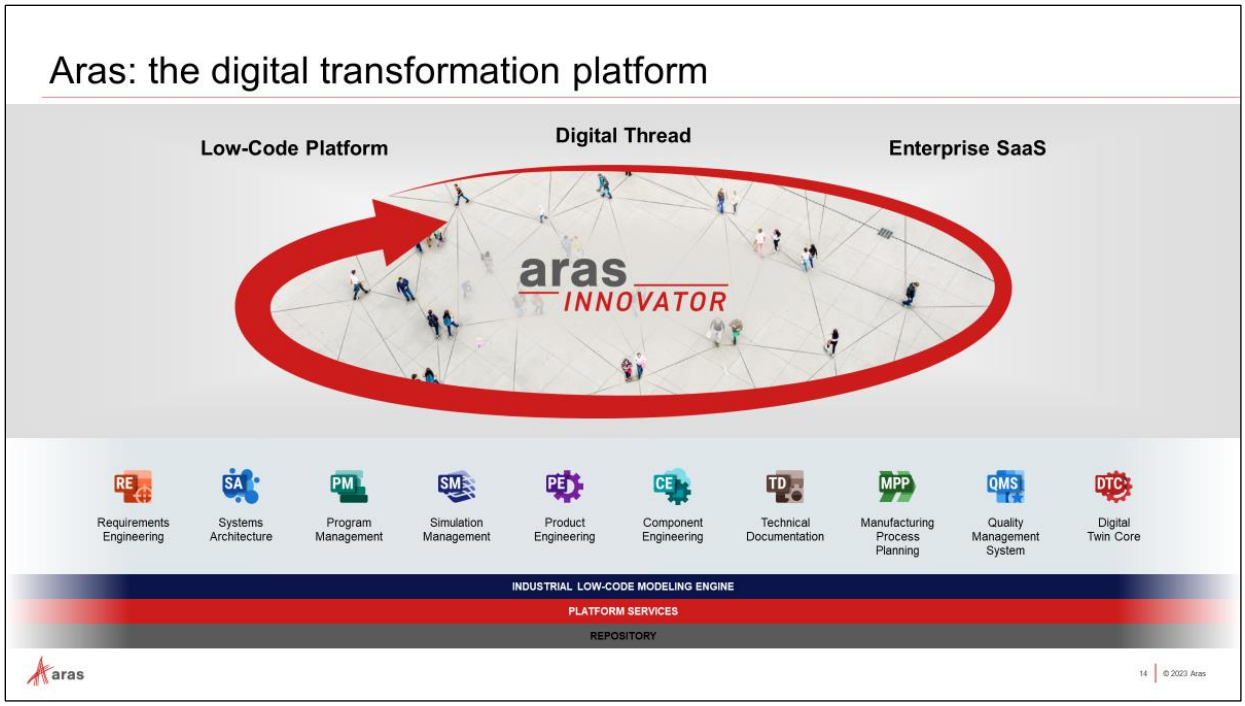
Where to go for Help

There are numerous active blogs and forums available from https://community.aras.com that answer many questions. Documentation sets can be found at http://www.aras.com/support/documentation/. Additional Aras online documentation is available for all products and releases by logging into the Subscriber Portal: https://www.aras.com/community/subscriber-portal.

All Aras Innovator browser clients also support an online, searchable Help file.

Note

Additional resources and tools are available with an Aras Innovator subscription.



Aras Platform Overview

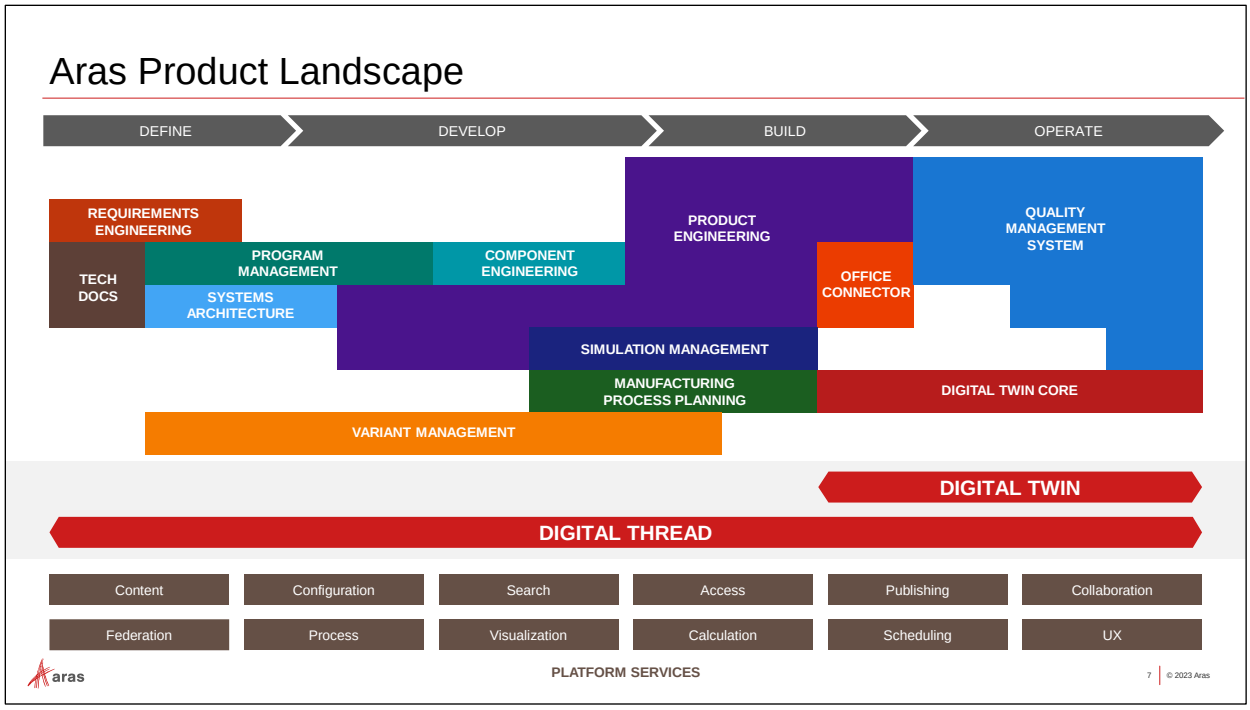
Model-based technology is flexible. Adaptive data models, process models and business rules fit your business now and in the future. Instead of the traditional approach - defining an object model hard coded into the services – Aras® separates the two. The result is a run-time web application framework consisting of loosely coupled, federated web services optimized for performance and scalability, which form a service-oriented architecture (SOA).

The core of the Aras platform is a suite of services that enable PLM behaviors on any type of data. Platform services include things like search, access control, workflow, lifecycle states, version control and File Vaulting. These core services are built to be highly secure and extremely scalable.

Then we use meta-data templates to describe everything: application functionality, business logic, data schema, screens, workflows, etc. The meta-data templates define the objects in the system and those objects then subscribe to the services that they need. We call it a model-based PLM and it is the basis for creating flexible, scalable, upgradeable applications using the modeling engine.

The modeling engine describes the data, processes, logic and user interface needed to build a PLM application. Aras builds its applications using the modeling engine. You can use our applications as-is, adapt them to your needs, or create new ones using those same modeling tools. Additionally, you can further extend capabilities with low code methods including integration with your own or other applications. Applications built with the modeling engine are upgradeable no matter how much customization has been made. (Details and demonstrations are available at Aras.com.)

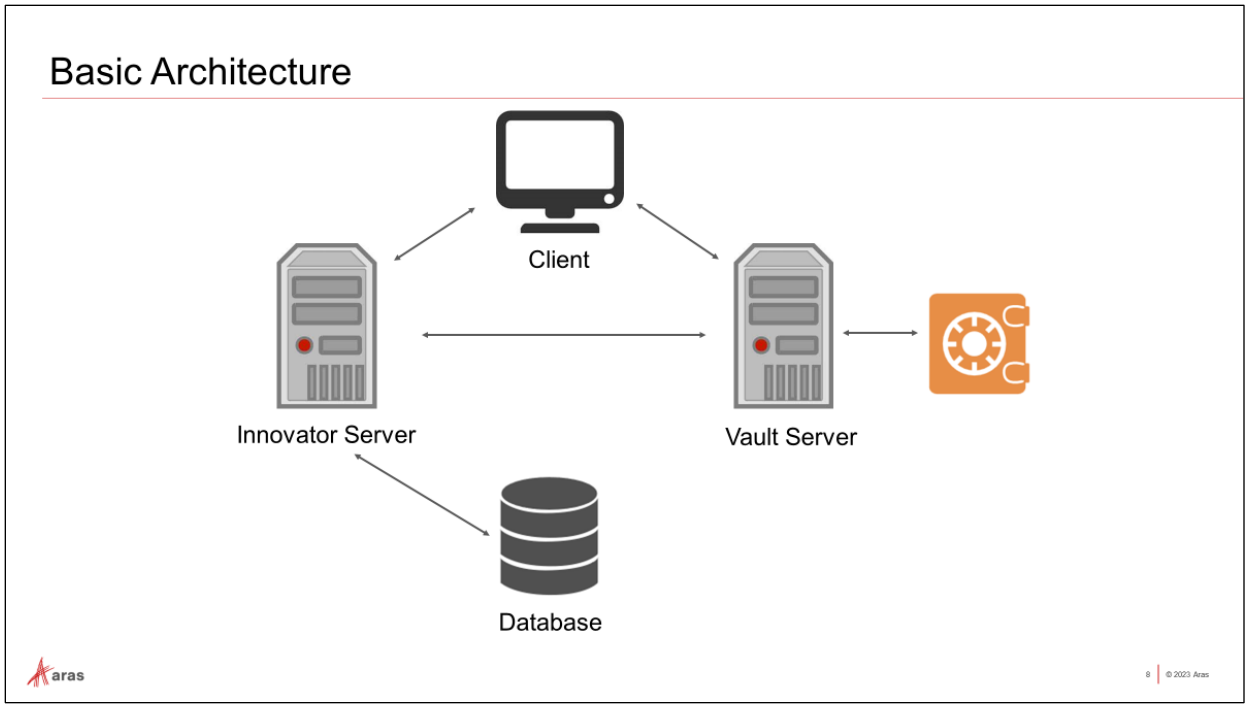
Users can connect the platform via various clients, including web and mobile interfaces. External systems can be integrated via Connectors, using the open Aras APIs and modeling techniques.



Aras Product Landscape

The Application layer represents specific business functions commonly used by a business to support their operations. These applications are built using the modeling engine and platform services.

- Subscribers of the Aras platform not only have access to all the existing applications, but they also have access to changing the application through the modeling engine to meet their needs.
- A critical advantage of the Aras Platform is that all the applications existing today or being developed in the future (by Aras or your organization) will use the same data model across the end-to-end processes.
- Aras manages growing application base by ensuring their single data model is completely aligned across applications.



Basic Architecture

Aras Innovator is an Internet application framework based on the Microsoft .NET platform. The basic components include:

Aras Innovator Client

Aras Innovator uses a web-based browser interface which requires very few client resources. Aras Innovator client runs in Microsoft Edge, Mozilla Firefox, or Google Chrome browsers.

Aras Innovator Server

Running as a Microsoft .NET application on Microsoft IIS 8 or higher requires the Microsoft Windows Server platform.

Database

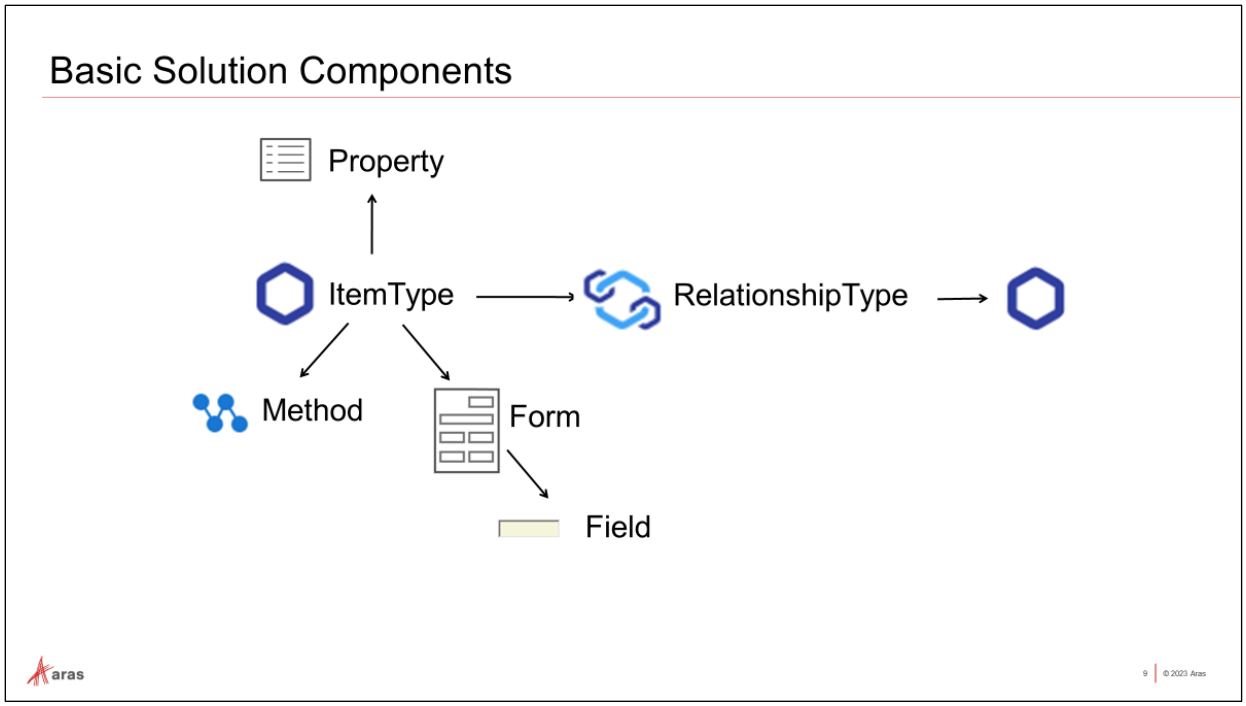
All configuration rules and code, as well as solution business objects are stored in the Microsoft SQL Server database.

Vault Server

A separate server application maintains information about files that are linked to objects in the SQL Server database.

Vault(s)

A file directory location made known to the vault server to store physical files.



Basic Solution Components

ItemType

An ItemType is a template that defines the structure of a business object in the database. You create ItemTypes in Aras Innovator which define how data will be stored in the database and how objects created from this type will behave. An ItemType has a corresponding table in the database that stores instances of that ItemType.

Property

A Property is a piece of information tracked about an Item. Each Property has a name, a label, a data type, and several other settings to define its behavior. A Property corresponds to a column in the ItemType's table.

RelationshipType

RelationshipTypes define which ItemTypes are related to one another, and the behaviors of the relationship. RelationshipTypes have a Source ItemType, an optional Related ItemType and a Relationship ItemType.

Form

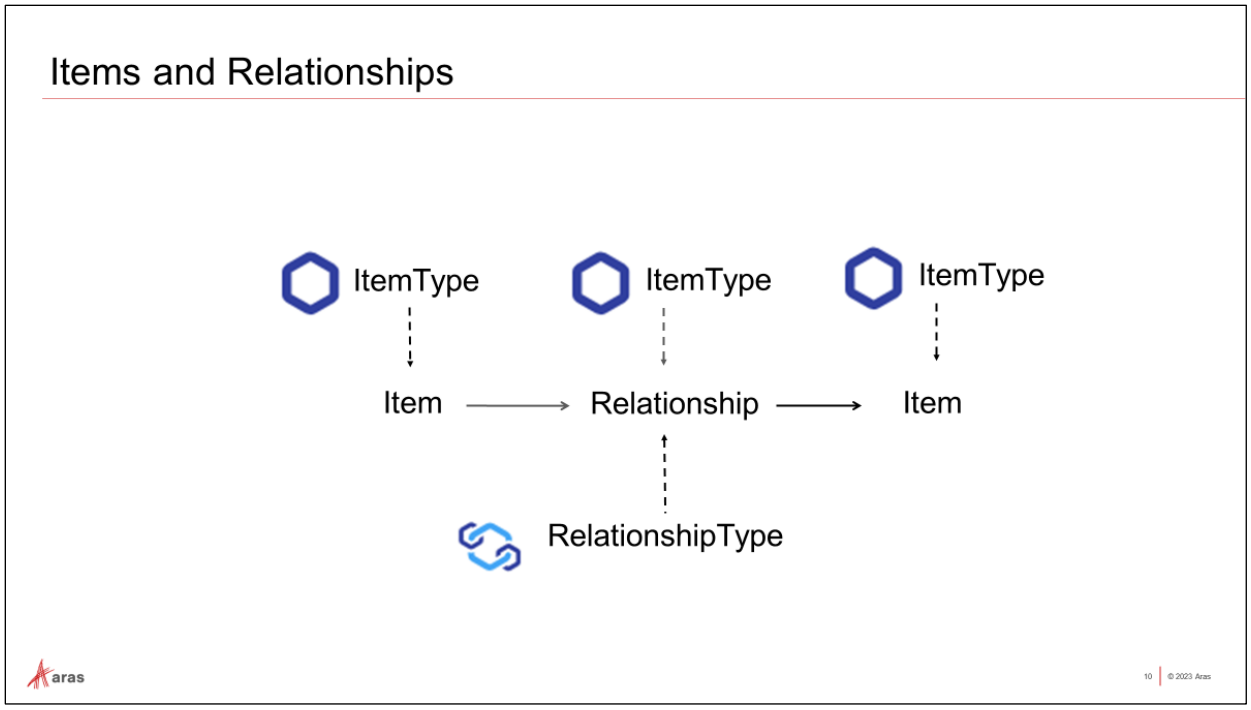
A Form allows a user to interface with an Item and is often used to view and edit instances of an ItemType. Forms are configured using a drag and drop utility, stored in the database, and rendered dynamically.

Field

A Field is used to display information on a Form. Fields may be, but are not required to be, bound to a property.

Method

A method is an Aras Innovator ItemType containing executable .NET language code, C#, VB, JavaScript. Actions and Events trigger the execution of Methods. Methods determine the behavior of an Item.



Items and Relationships

An instance of an ItemType is an Item ; each Item has distinct property values, can also be described with a lifecycle state and be controlled by a workflow.

When configuring the data model, you define ItemTypes that allow your end users to create Items of that type.

For example, Aras Innovator contains a Part ItemType that allows users to create unique Part instances in the database.

Relationship

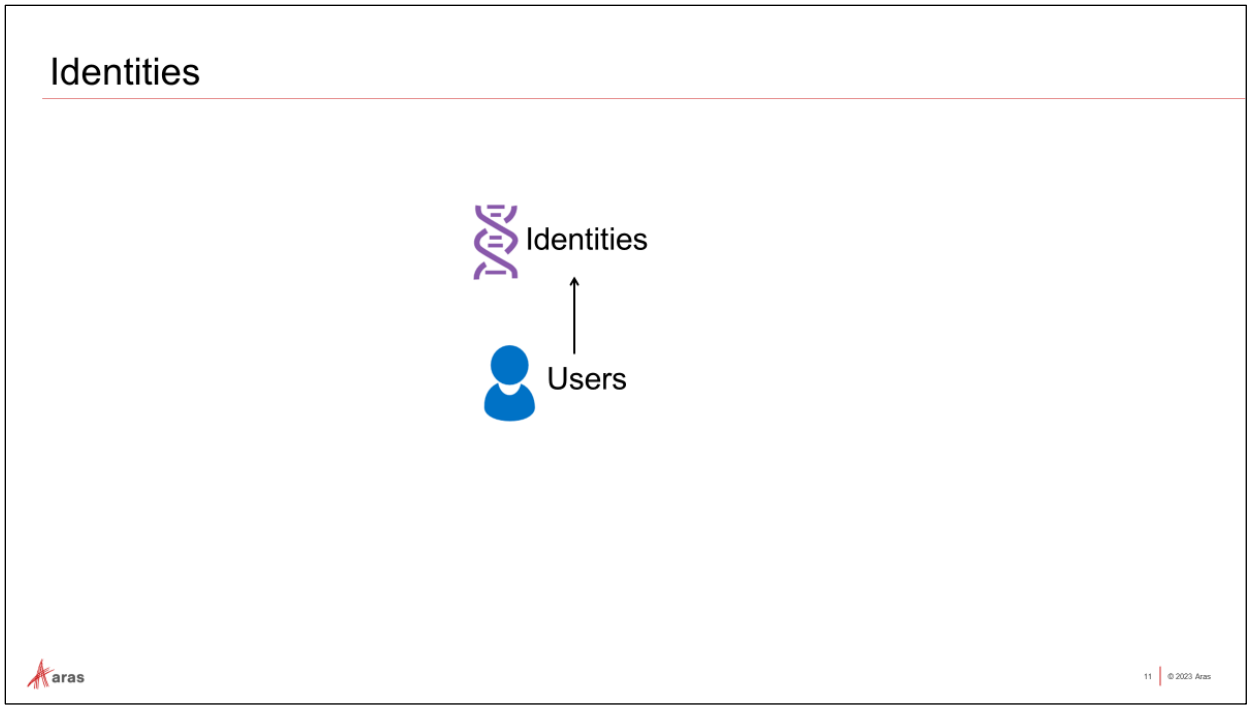
ItemTypes can be linked together to form relationships. When a Relationship is defined on an ItemType, two new Items are created: a Relationship ItemType and a RelationshipType. The new ItemType has a corresponding SQL table, to store information about the relationship. Properties can be added to describe the relationship. A new entry is added to the SQL table each time a related item is linked to the source item.

For example, a Part can be linked to a Document. A Document has a relationship to one or more files and is referred to as a "container". Both have entries in the Relationship ItemType in the database table. There is an entry in the Relationship ItemType in the database table that defines these relationships.

RelationshipType

RelationshipTypes define which ItemTypes are related to one another, and the behaviors and constraints of the relationship.

For example, if the business process rules dictate that every part must be linked to a CAD document item (e.g., CAD drawing), this would be set as a minimum of 1 must always exist.



Identities, Roles, and Users

Aras Innovator contains a rich, highly configurable security model that allows you to control who has access to the system and what they can do once authenticated.

Users

To access Aras Innovator, a User must be defined in the system with unique credentials, including username and password.

Identities

Identities form the basis for the Aras Innovator security model. Identities are hierarchical, and may represent an individual, a group, a group of groups, a role, a team, and so on.

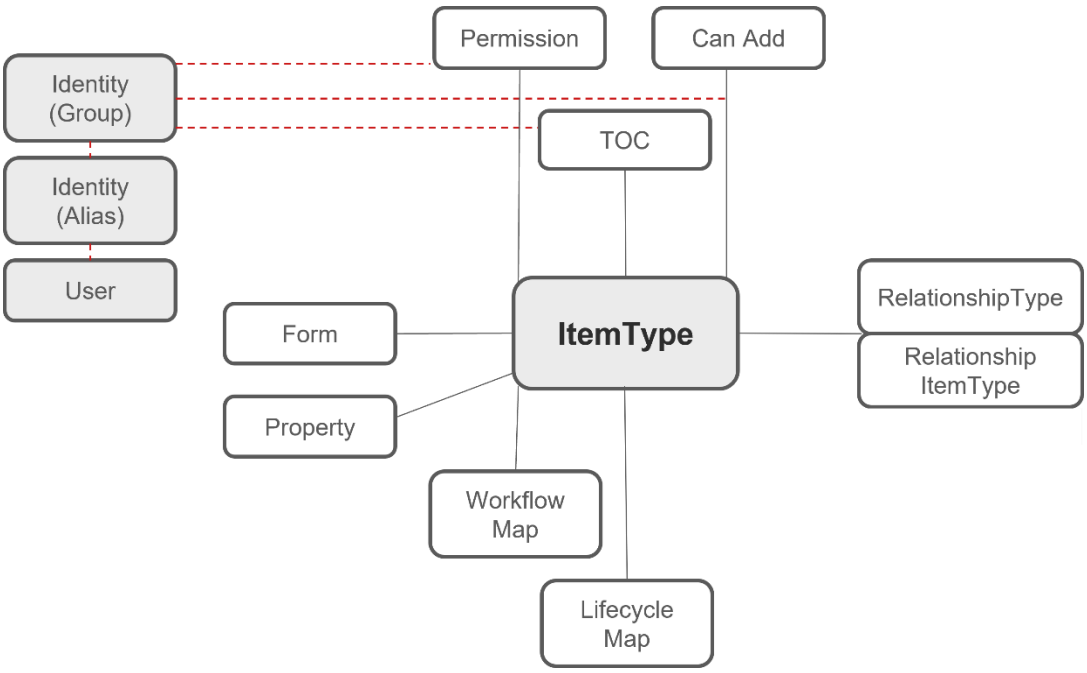
Unit 2 Users and Identities

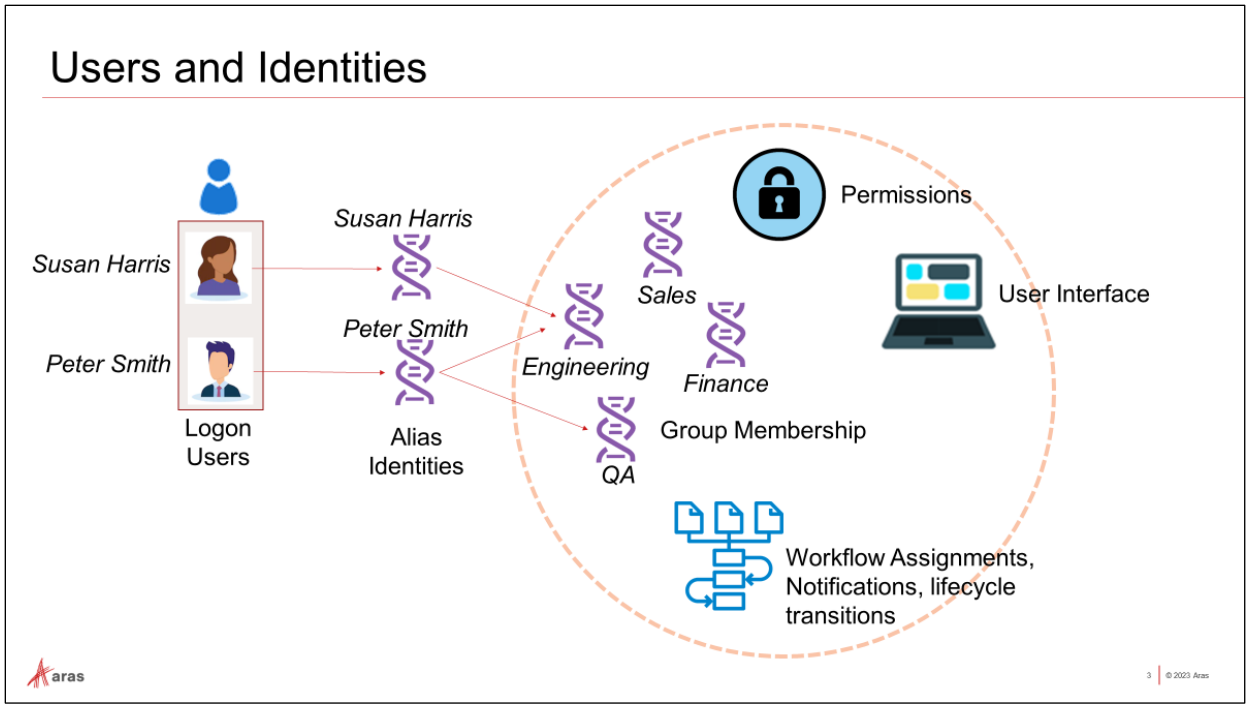
Overview

In this unit, you create Users and Identities in Aras Innovator.

Objectives

- Define Users and Identities
- Assign Members to Group Identities
- View Report on Identity Membership
- View System Identities





Users and Identities

The Aras Innovator data model provides both **User** and **Identity** ItemTypes. Each provides important functionality in the system:

Users

- The User ItemType manages login name and password credentials for logging into Aras Innovator, as well as employee contact information.

Identities

- Access control, user interface selection, workflow assignments, and many other system functions use Identities (vs. Users).

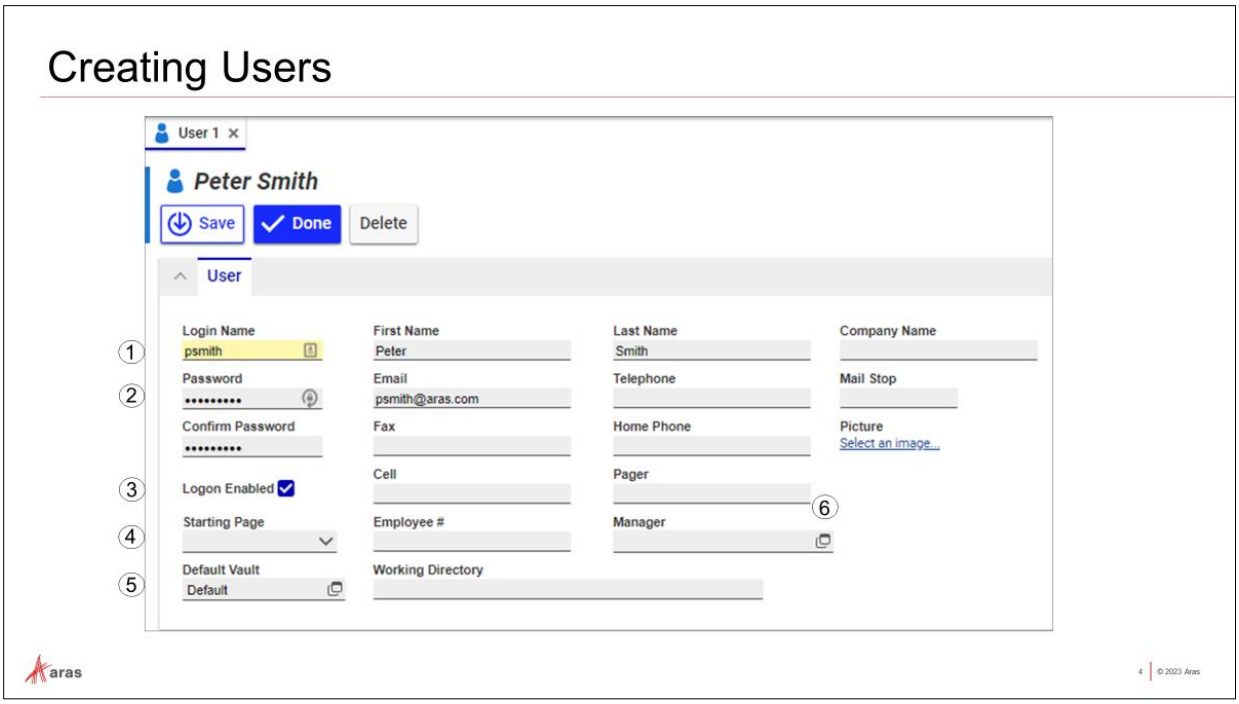
An Identity item can represent a user alias, a group or a role within system operations.

Identities

Whenever a User item is created, the Innovator Server automatically generates a corresponding system record known as an **Alias Identity**. Every User gets an Alias Identity to represent them as actors in system operations. Alias identities can be part of several group identities to inherit accesses.

Identities are used in:

- ItemType Permissions: what actions may be performed on an Item (view, modify, etc.).
- User Interface selection: the User Interface can differ depending on the access.
- Rights to create new item instances.
- Workflow Assignments, such as an approval process in a workflow step.
- Lifecycle Promotions, etc.



Creating Users

A user account is created for any person who will be logging into the Aras Innovator server. The User contains the credentials and additional profile information.

The two required fields are Login Name and specifying a Default Vault. All the remaining fields are optional, but frequently set to provide additional profile information. All login names must be unique within an Aras Innovator installation.

The following fields define a User item:

①	**Login Name**	Must be unique and is supplied by the user at log on.
②	**Password**	Is encrypted in the database and is required for a user to be authenticated through the Logon screen of the Aras Innovator Client.
③	**Logon Enabled**	Required to be checked for the User to be “active” in the system. If a User leaves an organization, this setting should be disabled. Removing the User record is normally not possible, because Aras Innovator keeps a history of Items that each user creates and edits.
④	**Starting Page**	The default starting page for the user (Configurable InBasket, My Discussions, & My Reports).
⑤	**Default Vault**	The Default Vault assigned to the user for File Check In.
⑥	**Manager**	The user specified as the manager of the user. There is no specific functionality attached to the Manager.

Try it … Create a New User

As a standard practice and for traceability purpose, users should be logging in with their unique login accounts and not using a generic user login. Any users who are responsible for System Administrative duties should be added to the Administrators identity. There are some default system users that exist as part of the installation, but their use should be limited and tightly controlled by the Administrators.

In this exercise, you will create a login for yourself to use throughout this course. Make sure you are in the correct database as specified by your instructor.

1. Login to your assigned database as 'Admin' with the password provided by your instructor.
2. Navigate to Administration > Users.
3. Click Create New User.
4. Complete the following fields with your desired user information (*Please use a login name and password that you will easily remember for the remainder of the class*):

Login Name	**First Name**	**Last Name**	**Password**	**Email**	**Logon Enabled?**	**Vault**
<login>	<first name>	< last name>	<password>	<email>	Yes	Default

An example is shown below.

Login Name	**First Name**	**Last Name**	**Password**	**Email**	**Logon Enabled**	**Vault**
mriley	Morgan	Riley	innovator	mriley@aras.com	Yes	Default

5. Click Done.

Try it … Create Engineering Users

We will create additional users that will be needed throughout this course and for the lab exercise at the end of this unit as well.

1. Use the following values to create 3 new users who will be allowed to work with Requests.

Login Name	**First Name**	**Last Name**	**Password**	**Email**	**Logon Enabled**	**Starting Page**
psmith	Peter	Smith	innovator	psmith@aras.com	Yes	
sharris	Susan	Harris	innovator	sharris@aras.com	Yes	My Discussion
jyoung	John	Young	innovator	jyoung@aras.com	Yes	Configurable InBasket

2. Be sure to check Logon Enabled for each of your new users.
3. Click Done for each user.

Defining Identities

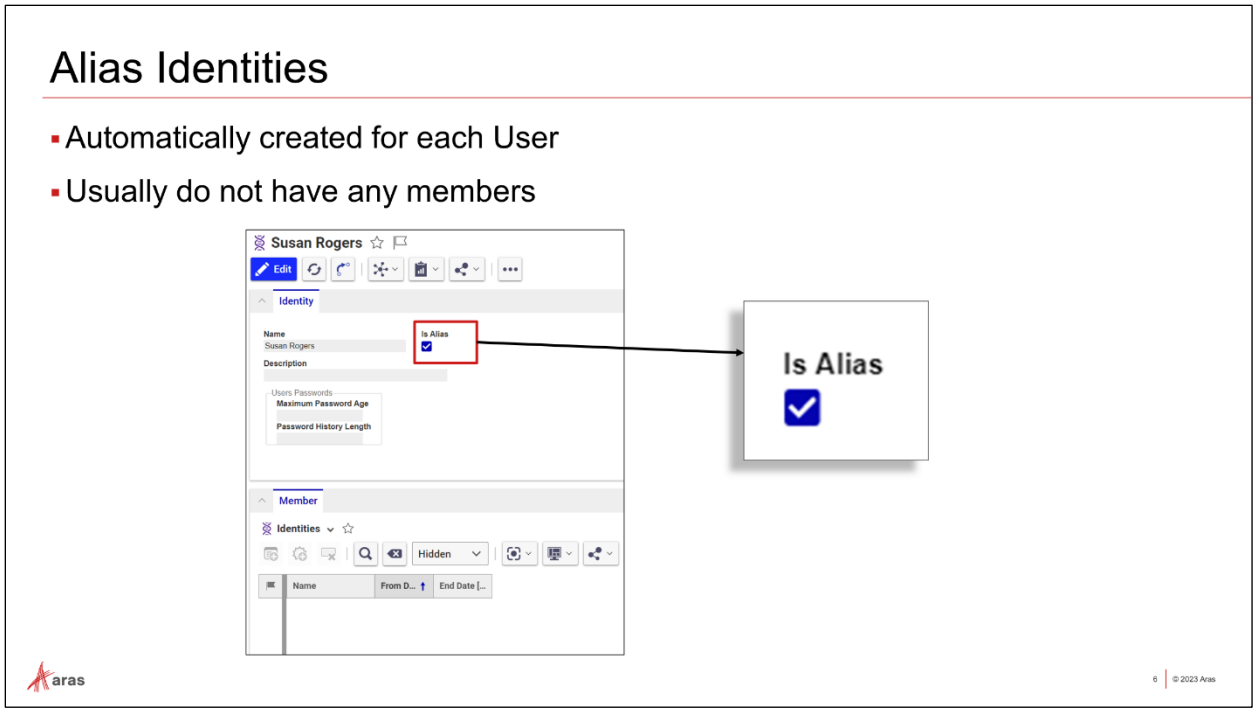
- Identities represent:
  - An individual (Alias Identities)
  - A group of individuals (Group Identity)
  - A group of groups (Group Identity)
  - Roles like Creator, Owner, Manager (Built-In Identity)

aras

5. | © 2023 Aras

Defining Identities

Aras Innovator uses a single Identity ItemType to provide uniform schema for important system functions including access control, workflow assignments, etc.



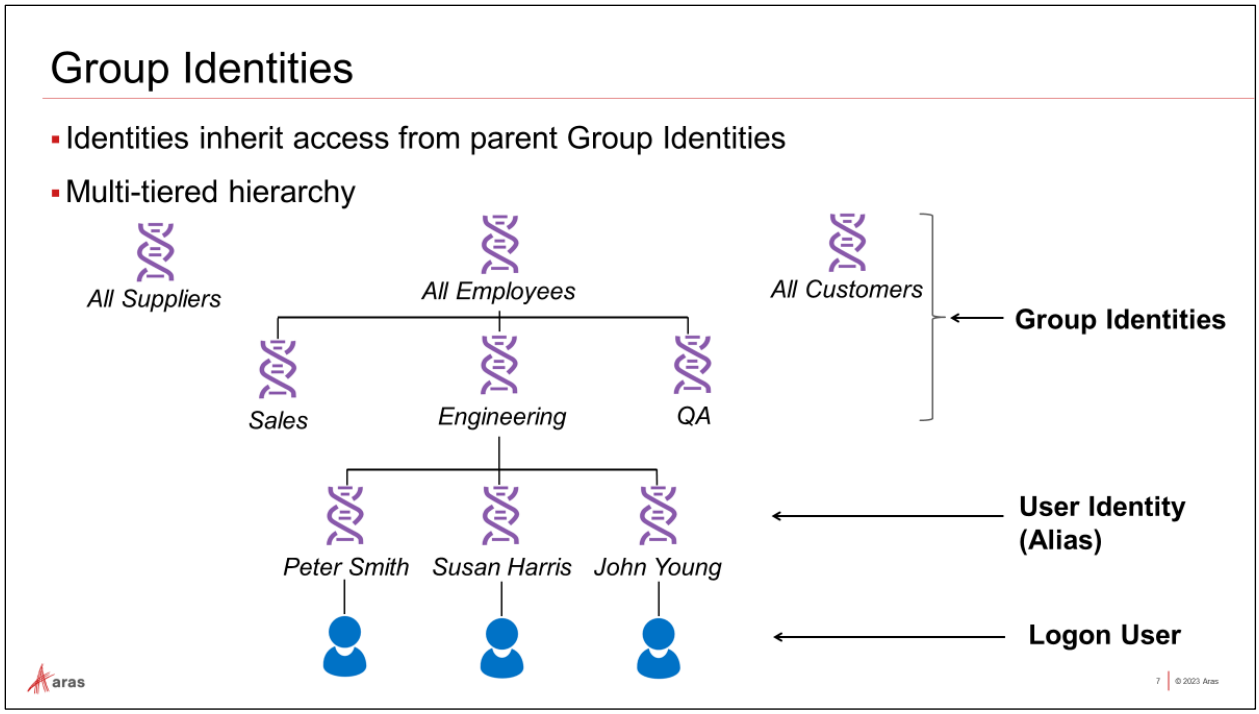
Alias Identities

The Alias Identities are linked it to the User using an Alias Relationship. The Alias Identities are designated by the 'Is Alias' Boolean property being set to true. The relationship of users to identities controls access to items, workflow and project assignments, and other system functionality.

An alias identity represents an individual, so it usually does not contain any members.

Try it … Open the Alias Identity of your personal User Item

1. Come back to your personal User created previously. Open the Aliases tab. If need be, maximize the Accordion window.
2. Open the Alias Identity that was automatically created by the system and notice that the *Is Alias* Boolean is set to true.



Group Identities

Group Identities define a list of User Identities that share the same access. Group Identities are used to represent departments, roles, etc. Group Identities can also be assigned membership in other Group Identities, forming a hierarchy.

With this multi-tiered hierarchy access, all members of a Group Identity inherit the access privileges of the Group Identity they belong to. Members are assigned through the Member relationship tab in the Group Identity. Identities may belong to more than one group identity, in which case the user's access or permissions are always cumulative.

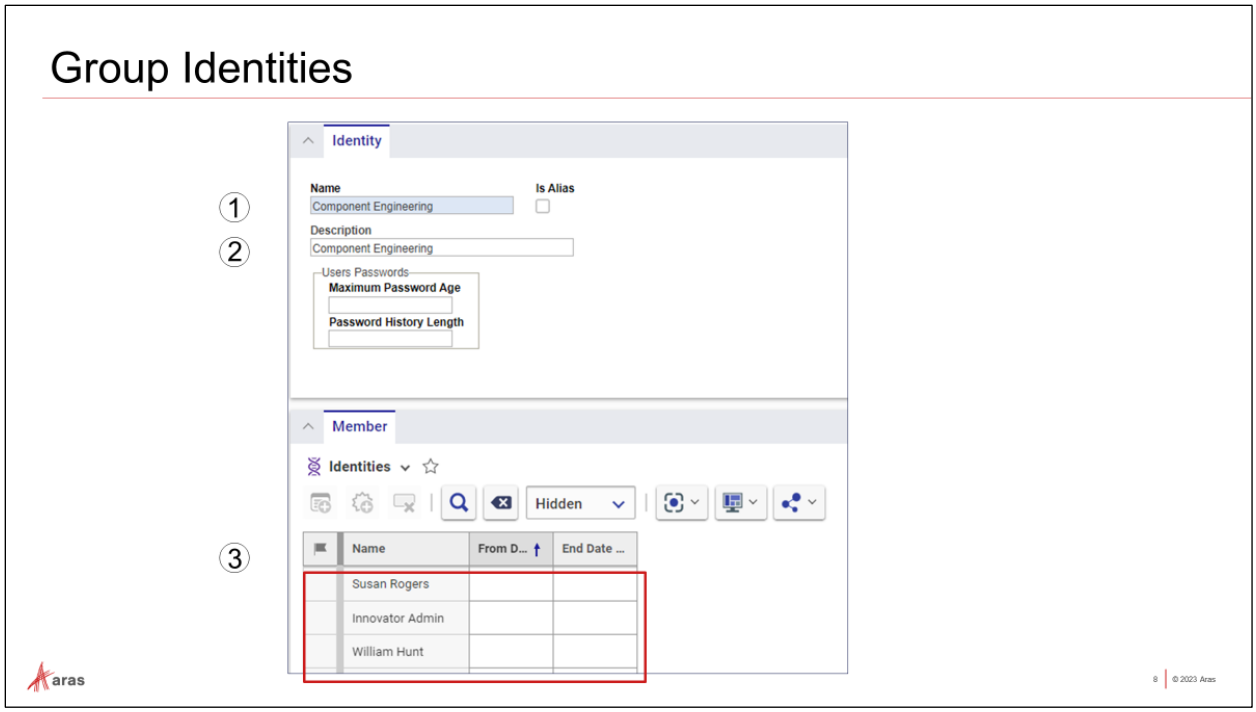
In the example above, Peter Smith is an Employee who belongs to the Engineering Identity. Rights granted to both the All Employees group and the Engineering group cascade down to Peter.

A Group Identity may contain as many members as necessary. Once the company structure is in place, the top internal department Identities should be added as members to the All Employees Identity. It creates a distinguishable Identity between Employees and external groups such as All Suppliers and All Customers.

Tips

When planning Users and Identities, think “top down” like a company organization chart.

Use Group or Role Identities for workflow assignments and lifecycle promotions; they are more flexible than Alias Identities. It is easier to change group membership than individual Identity permissions.



Group Identities

The Group Identities members are maintained in Member relationship tab. The 'Is Alias' Boolean is unchecked for Group Identities.

Note

Group Identities are not the only Identities with 'Is Alias' unchecked: System Identities have the same.

The following fields define a Group Identity:

①	**Name**	The Name of the Group (typically a role in the organization).
②	**Description**	The Details about the group.
③	**Member Name**	The Member tab lists all members of this group Identity.

Both *From Date* and *End Dates* are deprecated features and no longer supported.

Try it … Add a User to a Group

Your new user will need to act as an administrator and employee during this class. We will now add your new Alias Identity to the following Group Identities: Administrators, All Employees and Component Engineering.

1. Navigate to Administration > Identities.
2. Search for the **Administrators** Identity Group and open it for editing.
3. In the Member relationship tab, click the Add Identities button.
4. Search for and select the new Alias Identity you created, then click OK.
5. Click Done to save and unlock the Administrators Group Identity.

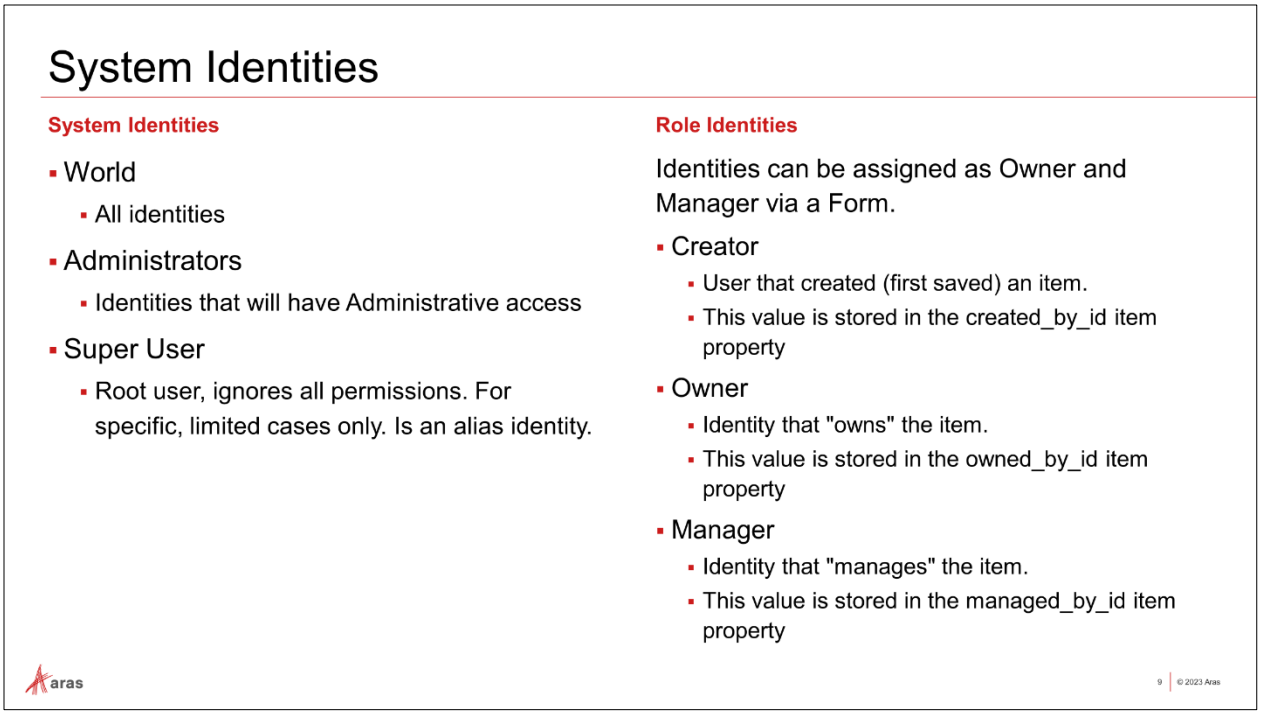
6. Search for the **All Employees** Identity in the search grid and open it for editing.
7. Add the **Administrators** group Identity as a member, then click Done.
8. Search for the **Component Engineering** Identity in the search grid and open it for editing.
9. Add the new Alias Identity of the personal User you created, then click Done.

   We now add administrators as members of the Report Authors identity, to allow them to create reports in the future.
10. Search for the **Report Authors** Identity in the search grid and open it for editing.
11. Add the **Administrators** group Identity as a member, then click Done.

Try it … Modify a Group Identity

In this exercise, we modify an Engineering identity that is also used throughout this course and is required for Lab exercises.

1. Navigate to Administration > Identities and open the *Engineering* Identity for editing.
2. Add these new members:
   - Peter Smith
   - Susan Harris
   - John Young
3. Verify that Diane Prescott is also a member. Click Done.
4. Add the **Engineering** Group Identity as a member of the existing **All Employees** Identity.
5. Click Done.



System Identities

There are several System Identities in Aras Innovator that are reserved for specific purposes, and cannot be deleted or changed.

The role identities like Owner and Manager can be granted specific privileges in a security policy ; when their related property is placed on a Form, an individual or group can be chosen when the item is created or edited.

World

Represents all identities in the system. Every user in Aras Innovator belongs to the World.

Administrators

Allows an Identity to have administrative access to Aras. This includes but is not limited to access to the Administration templates in the TOC and the ability to create further group identities with other administrative rights, and the ability to break an Item Claim.

Creator

This identity represents the User that created (first saved) an item. When an item is saved, the *created_by_id* System Property on the item is updated to point to the User creating the item (this value cannot be changed).

Owner

This Identity represents the value of the *owned_by_id* System Property on an item. Often it is useful to allow a Creator to be able to specify another individual (or group) that will have some interaction with this item later (e.g., lifecycle promotion, security, and/or workflow assignment).

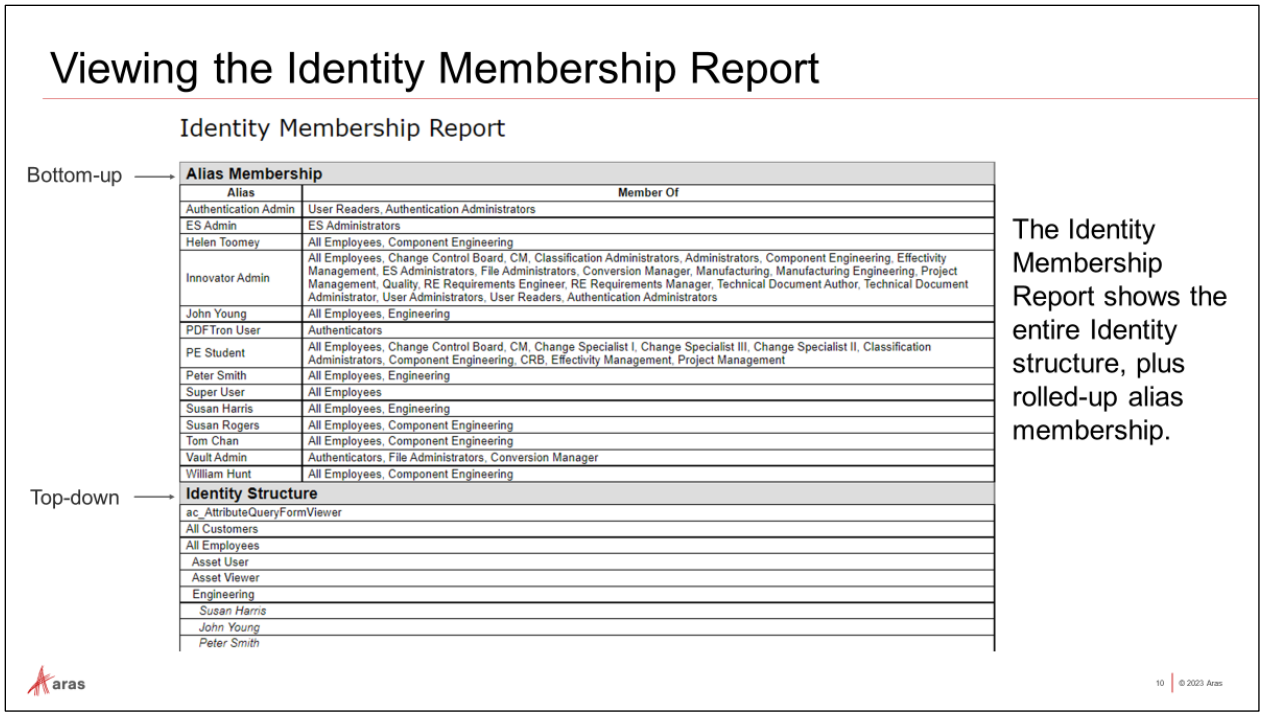
As an example, the Owner identity is used by a Part item to allow assignment of an Assigned Creator, or the identity responsible for the technical content of the item.

Manager

This identity represents the value of the *managed_by_id* System Property on an item. It can get review assignments on Items.

Super User

The Super User is a special user identified as the “root” user of the system. The “Super User” is defined by the system to ignore all Permission settings; thereby, having access to all Items independent of their Permissions. You should use this special Super User only in limited cases.



Alias Membership

Alias	Member Of
Authentication Admin	User Readers, Authentication Administrators
ES Admin	ES Administrators
Helen Toomey	All Employees, Component Engineering
Innovator Admin	All Employees, Change Control Board, CM, Classification Administrators, Administrators, Component Engineering, Effectivity Management, ES Administrators, File Administrators, Conversion Manager, Manufacturing, Manufacturing Engineering, Project Management, Quality, RE Requirements Engineer, RE Requirements Manager, Technical Document Author, Technical Document Administrator, User Administrators, User Readers, Authentication Administrators
John Young	All Employees, Engineering
PDFTron User	Authenticators
PE Student	All Employees, Change Control Board, CM, Change Specialist I, Change Specialist III, Change Specialist II, Classification Administrators, Component Engineering, CRB, Effectivity Management, Project Management
Peter Smith	All Employees, Engineering
Super User	All Employees
Susan Harris	All Employees, Engineering
Susan Rogers	All Employees, Component Engineering
Tom Chan	All Employees, Component Engineering
Vault Admin	Authenticators, File Administrators, Conversion Manager
William Hunt	All Employees, Component Engineering

Identity Structure

ac_AttributeQueryFormViewer
All Customers
All Employees
Asset User
Asset Viewer
Engineering
Susan Harris
John Young
Peter Smith

Viewing the Identity Membership Report

The Identity structure may resemble an organization chart or may have product team structures. In some cases, the structure will have both functional and product associations.

The example shown is for a sample training database, so it is simple. Companies may have many levels of structure. This report is useful as a quick reference instead of traversing a deep hierarchy with a Where Used search.

Alias Membership

Displays all Alias identities in one column. For each alias identity, it displays all the associated identities it is a member of, in a column to the right.

Identity Structure

A reciprocal of the Alias Membership displays a hierarchy of identities from top Group Identity down to the Alias. Aliases in the Identity Structure are also shown in italics for easy recognition.

Try it … Run an Identity Membership Report

1. Navigate to Administration, right-click on Identities, and select Identity Membership.
2. Find Peter Smith, Susan Harris and John Young in the Identity Membership Report. Confirm that they belong to *Engineering* and *All Employees* shown in the Member Of column.
3. Logout.
4. Login again using your new Administrator Login Name.

Summary

In this unit, you learned how to create Users and Identities in Aras Innovator.

You should now be able to:

- Define Users and Identities
- Assign Members to Group Identities
- View Report on Identity Membership
- View System Identities

Review Questions

1. What is automatically created when you create a new User? What is the purpose?

2. What is the best way to see a Group Identity hierarchy that has been defined in the system?

3. What should be done if a User leaves the Organization?

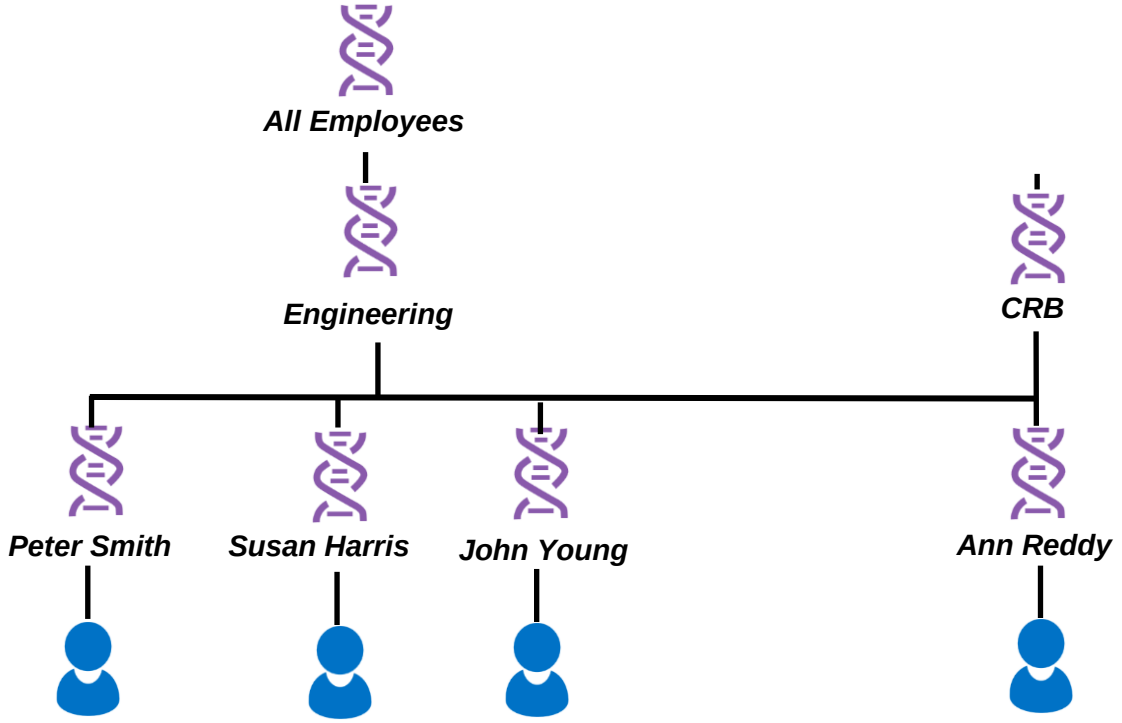
Lab Exercise

Goal

Create a new User and Identity based on a set of security requirements.

Scenario

You will be extending Identities so that when you are finished the organization looks like the following:



Preparations

1. You should already have added new members in the Engineering Identity: Peter Smith, Susan Harris, and John Young Identities. If not, you will need to add them by performing the previous try-its, before you start this lab exercise.

Users and Identities

1. Create a new User named *Ann Reddy* (Login Name: *areddy* with the *innovator* password), select Logon Enabled, click Done, and check to ensure the Alias Identity is also created.
2. Edit the Group Identity named CRB (Change Review Board) and add the Ann Reddy Identity as a member.
3. Also add the Ann Reddy Identity as a member of the Engineering Identity Group.
4. Run the Identity Membership report for Ann Reddy and check her group membership(s).

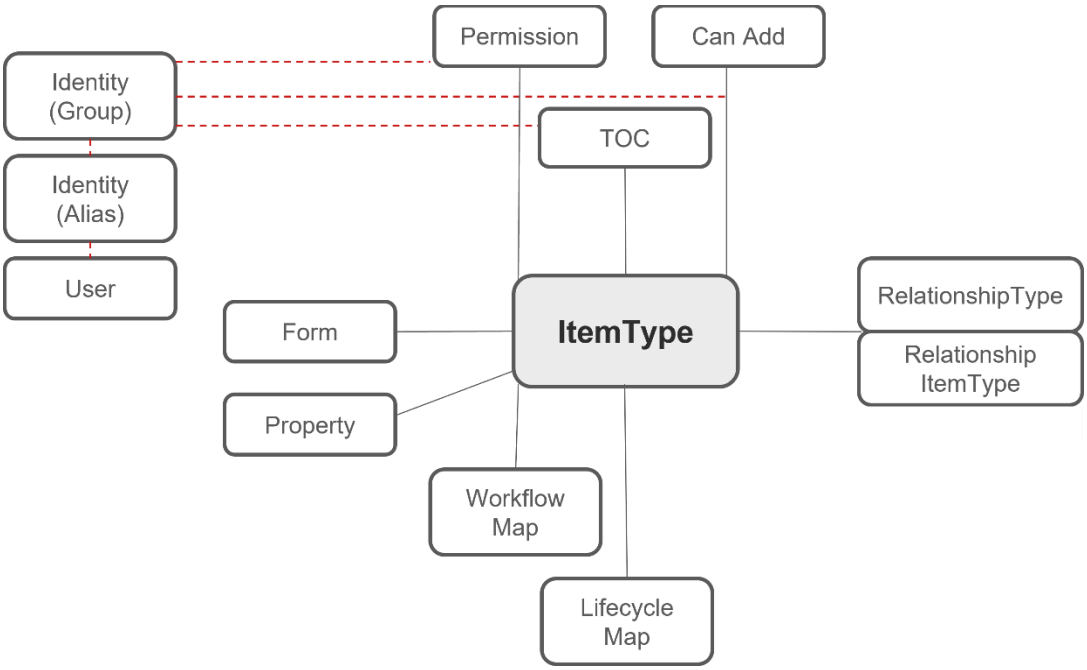
Unit 3 Exploring ItemTypes

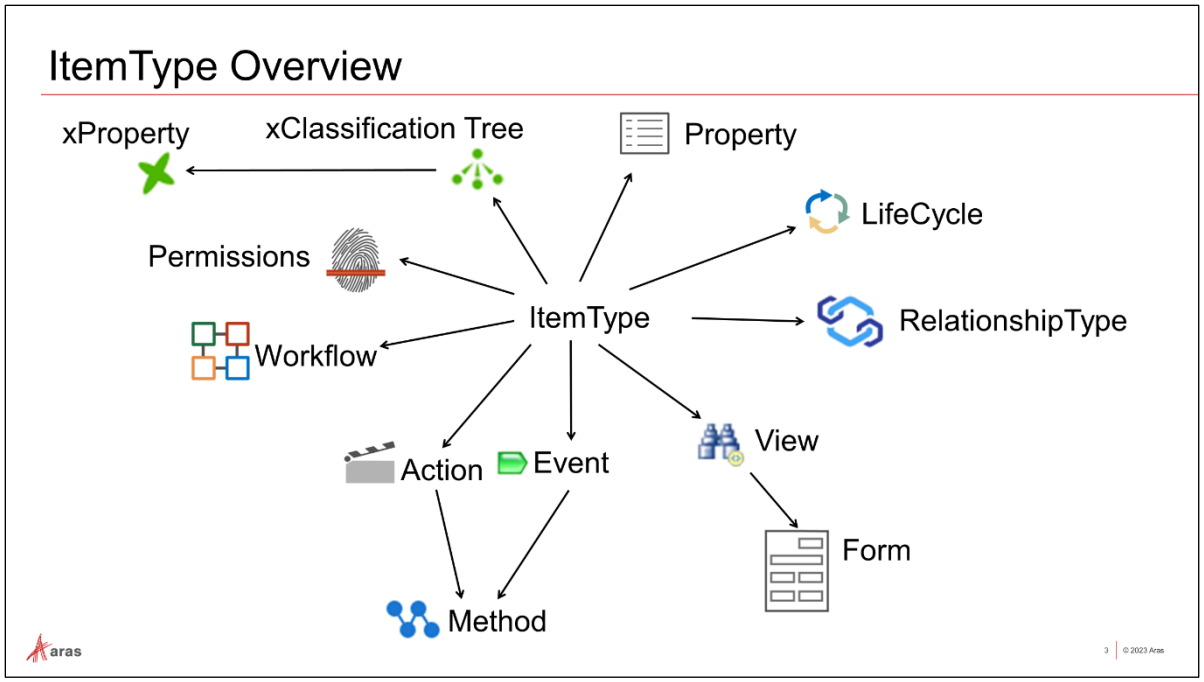
Overview

In this unit, you review the main configuration settings of an ItemType and learn how to create new ItemTypes. You also review several reports related to ItemType definitions.

Objectives

- Define the purpose of an ItemType
- Create a New ItemType
- Display ItemType Properties
- Run an ItemType Report

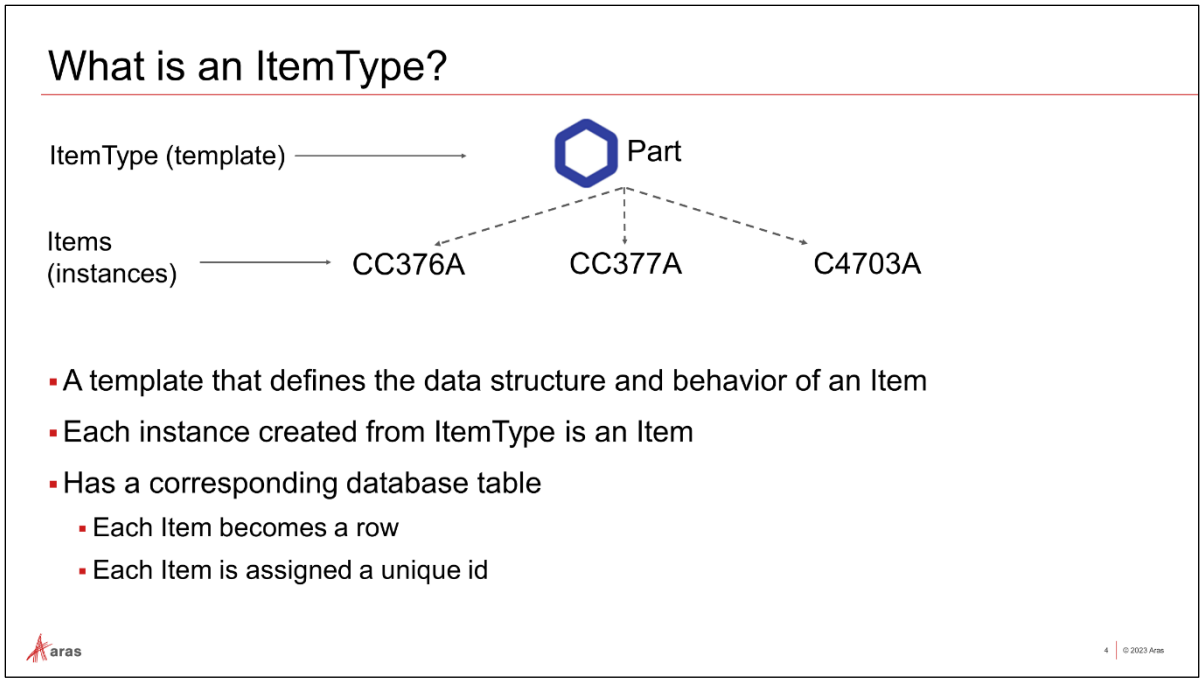




ItemType Overview

Almost everything in Aras Innovator is defined through an ItemType. ItemTypes define Properties, Forms, or Views available for the Item, Lifecycle Maps and Workflow Maps associated with the Item, Permissions, relationships, server and client Methods and Events that occur on the Item, and much more.

ItemTypes are the templates that define what kind of Items a User will create in a solution.



What is an ItemType?

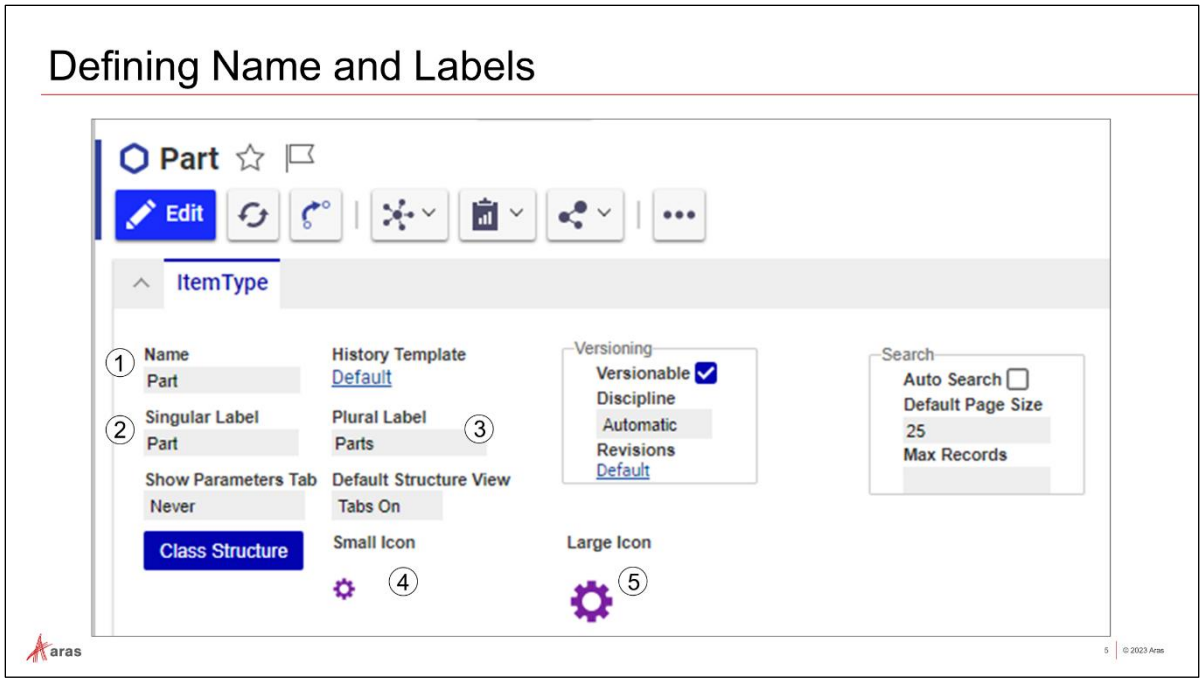
An ItemType is a template, a definition of Items created in the system. An item is an instance of an ItemType, a created business object and managed by Aras Innovator.

- An ItemType defines data structure and behavior, analogous to a class in object-oriented systems.
- ItemTypes are created by using a simple user interface and stored in the database.
- ItemType instances are called Items.
- ItemTypes are the basis for what makes Aras Innovator self-describing.
- ItemTypes have corresponding tables in the database.

Each Item that is created from an ItemType is assigned a unique 32-character identifier that is guaranteed to be unique for the enterprise (id property). This is commonly referred to as the GUID (Globally Unique Identifier).

Try it … Open an Existing Item Type Definition for Viewing

1. Login as an Administrator (your student login).
2. Navigate to Administration > ItemTypes in the Navigation Panel.
3. Search for and open the ItemType named **Part**.



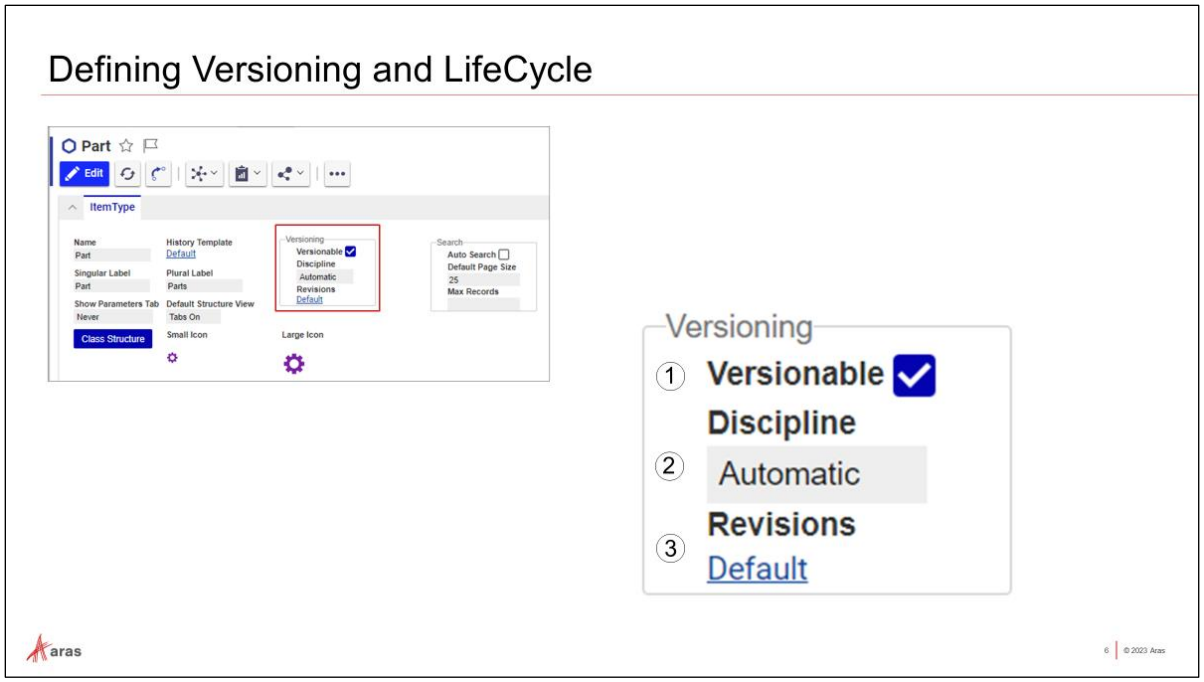
Defining Name and Labels

When you define an ItemType, you specify information about the type and the way Items created from it are used.

①	**Name**	ItemType Names may contain letters, numbers, spaces, and underscores. Since the ItemType name is stored as a table name in SQL Server, ItemType names can be just about anything. A SQL table name is created with all caps and spaces converted to underscores. In the above example, the table name is Part.
②	**Singular Label**	Labels are displayed throughout the system and come in either singular or plural forms. The singular label appears in the properties accordion header in an Item window, and in 'Create New…' buttons.
③	**Plural Label**	The system displays the plural label as a category of items in the TOC pane.
④	**Small Icon**	A small icon displays next to the item within the TOC and the Structure Browser.
⑤	**Large Icon**	The Large icon displays in the Form Layout as part of Item_info.

Users are free to use their own icons: Aras Innovator supports the Scalable Vector Graphic (SVG) format, which allows an image to be scaled to any size. Scalable Vector Graphic (SVG) files can be created and modified using many graphic editing tools including the open source Inkscape utility (www.inkscape.org).

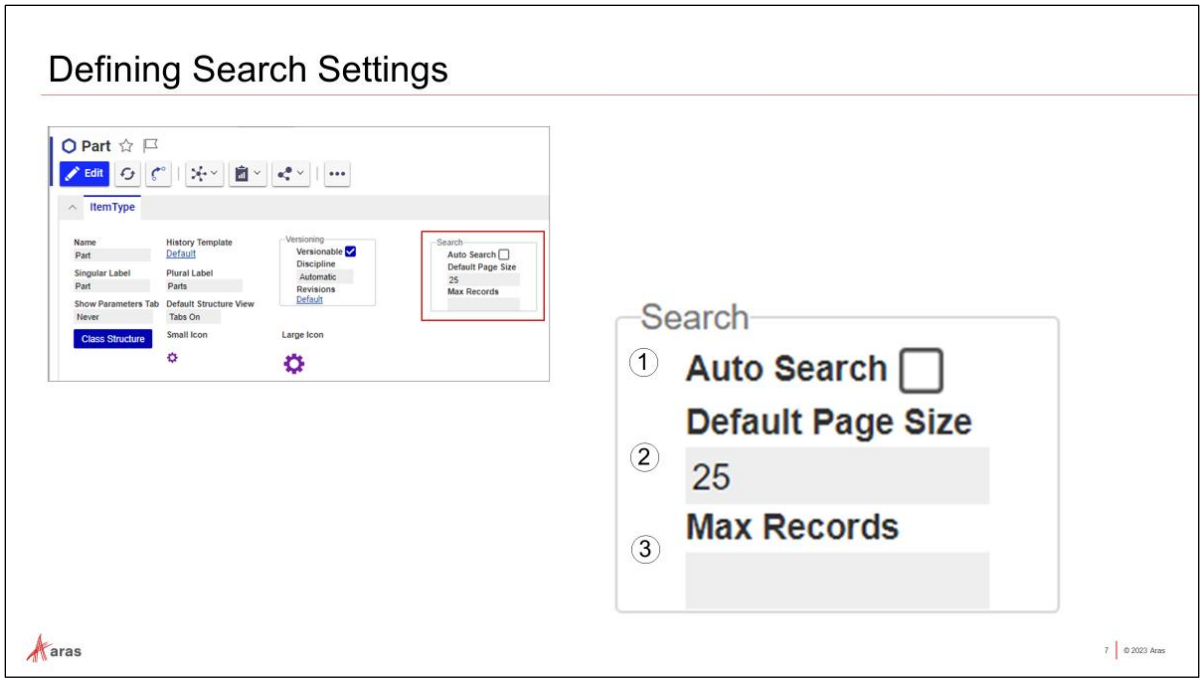
All SVG images provided by the Aras Innovator software are in a folder on the server: <Root>\Innovator\Client\images\. An additional folder is reserved for any custom SVG images created by customers: <Root>\Innovator\Client\customer\



Defining Versioning and Lifecycle Maps

Every Item in Aras Innovator can be promoted through a defined Lifecycle map. As an object is promoted through various states, its status and other characteristics may change. Items can also be versioned so that each time an editing change is made to an object, a new generation ("version") of the changes is made in the database. This allows you to view previous versions of an Item and make comparisons based on changes that have been made.

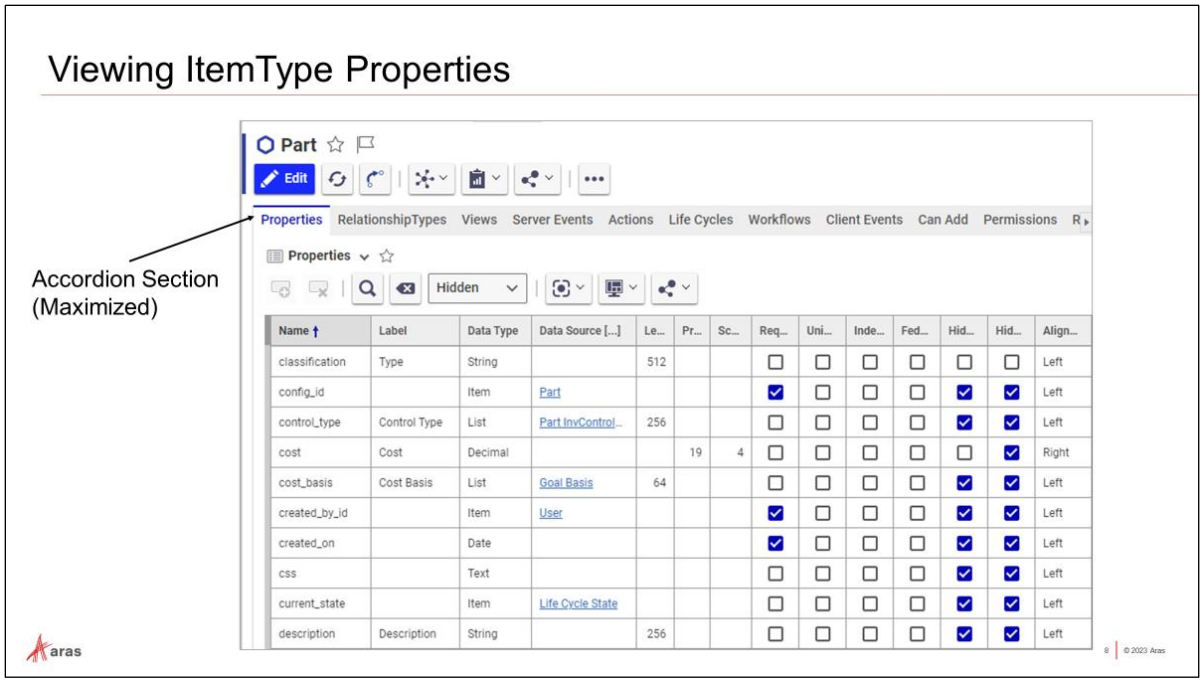
①	**Versionable**	When selected as an automatic process, the generation ("version") of an item is incremented each time the item is changed (i.e., edited, saved and unclaimed). This setting also works in combination with a Lifecycle Map that includes both a start state and release state. After an Item's state is set to Released, the Revision field increments to the next sequenced revision number after an edit and save. The Item is automatically set to the Lifecycle start state.
②	**Discipline**	**Automatic versioning** increments the generation of an Item each time it is edited and saved. For manual versioning, the item is only incremented upon explicit request by users. (Right-click on an Item in the search grid and select Create new Version from the menu.) The generation does not increment until a user chooses to manually version the Item.
③	**Revisions**	Revisions rely on a sequence object, which dictates how each revision will be labeled. The default sequence is alphabetically ordered from A – Z, minus the I and O. However, administrators can define both alphanumeric and numeric sequences to support their business rules.



Defining Search Settings

Settings can be overridden by users in various search grids, but you can set default values for Item searches.

①	**Auto Search**	When selected, the Aras Innovator client automatically displays search results upon opening a search grid. However, it can slow the loading of items since it assumes a full query search when a large number of items exist. It is recommended to set a Max Records and a Default Page Size to a practical value to produce quicker results.
②	**Default Page Size**	You can specify the default page size, which defines how many items will appear on the search grid. Often the default number is set to 25. If left blank, the page size equals the Max Records field. Note that users can override the page size setting when they perform a search.
③	**Max Records**	You can specify the default Max Records size. This limits the size of the returned data set, which can improve performance. Users can override the Max Records size when they perform a search.

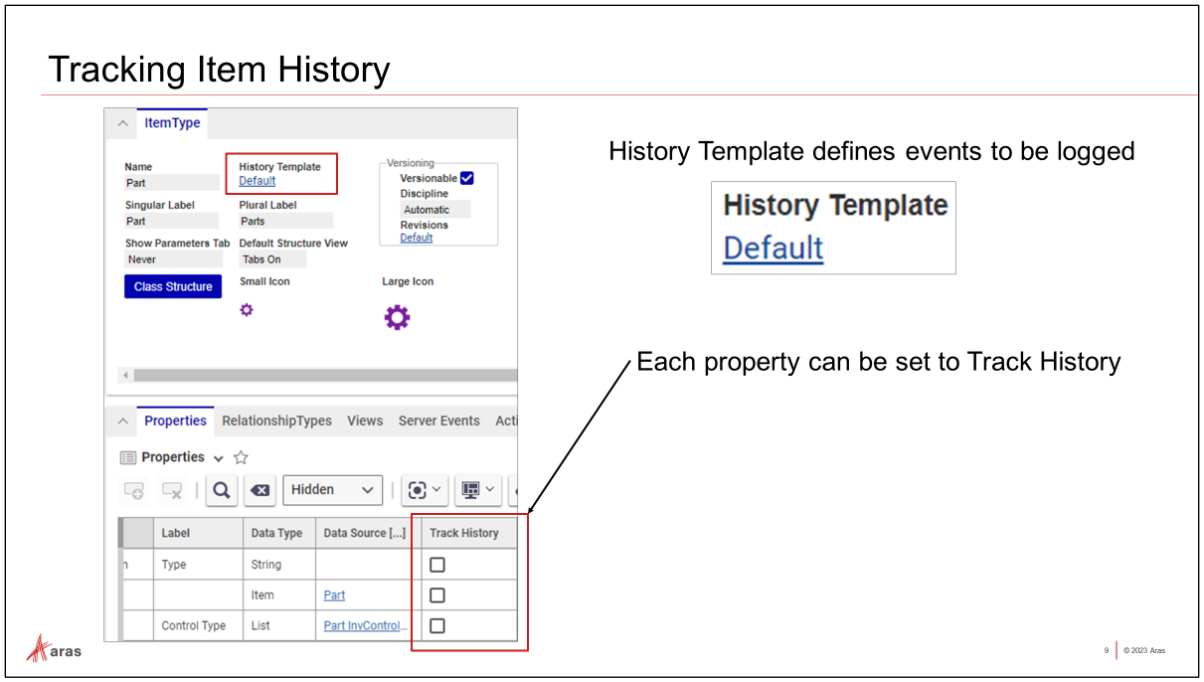


Viewing ItemType Properties

Properties hold pieces of information that are associated with an ItemType. Each new Item that is created from an ItemType in the system holds information relative to that instance of the Item. (For example: created_on holds the date and time an Item was created.) The Properties tab is a list of existing system and custom properties associated with this ItemType.

Try it … View ItemType Properties

1. Ensure the Part ItemType is open for viewing.
2. Click to select the Properties tab in the Relationships Accordion Section.



Tracking Item History

Aras Innovator provides the ability to track and record changes to an Item. A set of actions are used to determine which events are captured within a history log. History can be tracked at both the Item and Property levels. In the following example, when a user modifies an item's description property, the system not only tracks that the item was modified, but that the description property was changed (including the old and new values).

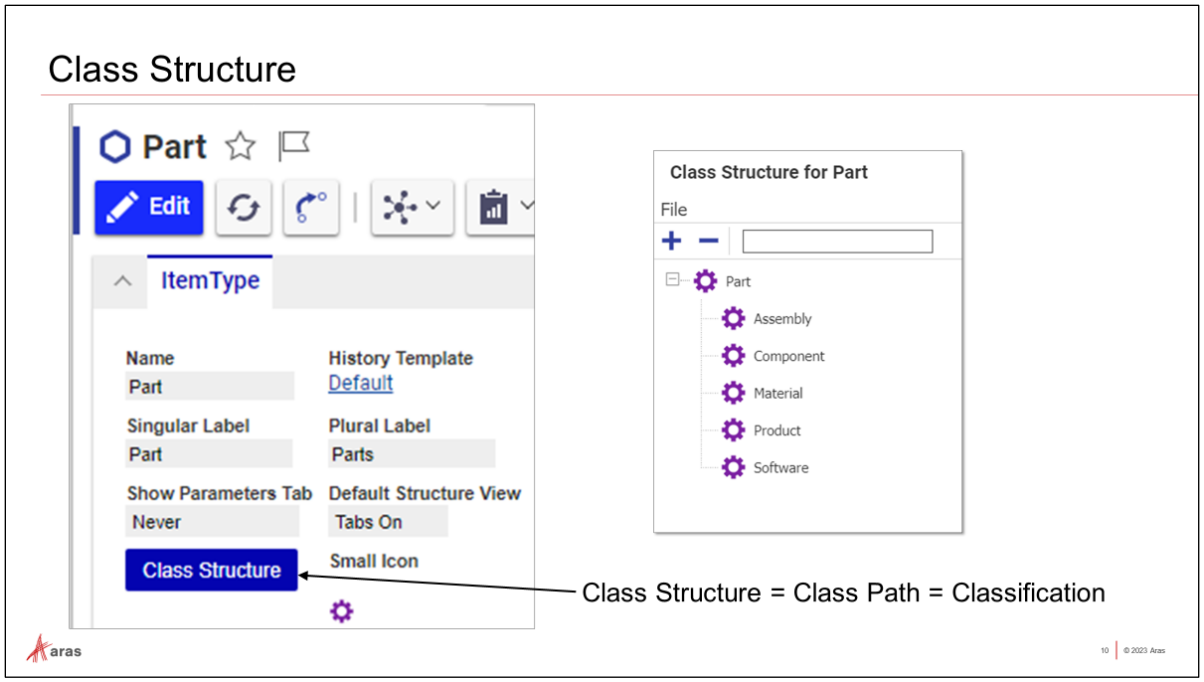
When [...]	Who [...]	Action	Comment [...]
1/11/2014 11:01...	Innovator Admin	Update	description: "Mobile Phone Model 1000" > "Mobile Phone Model 1011"

History Template

History Templates are used to define the set of actions that trigger a history record to be stored by the system. Aras Innovator provides a Default History Template that can be used as a reference for configuring the history of an ItemType. Alternatively, you can create custom History Templates.

Track History Setting for Properties

Track History may be enabled to record property value changes as part of the Item history. Set Track History on for each property on the ItemType that you want included in the history report. If the Track History option is deselected, no property changes are recorded for this ItemType.



Class Structure

ItemTypes may contain a structure of subclasses where each class can refer to its own specific properties, forms, and Lifecycle map.

A class structure can be many levels deep and arranged as a hierarchy depending on the complexity required. Also, this Class Structure is ItemType specific. The assignment of a standard classification to an Item can only be selected by navigating to the lowest level.

Classification allows you to share properties among subclasses. If you define a property at a parent level class, all subclasses will inherit those properties.

In the examples above, the Part ItemType has a defined class structure. Because each of the subclasses share common attributes, it can be more efficient to use class structure versus creating entirely new ItemTypes to differentiate different types of Parts.

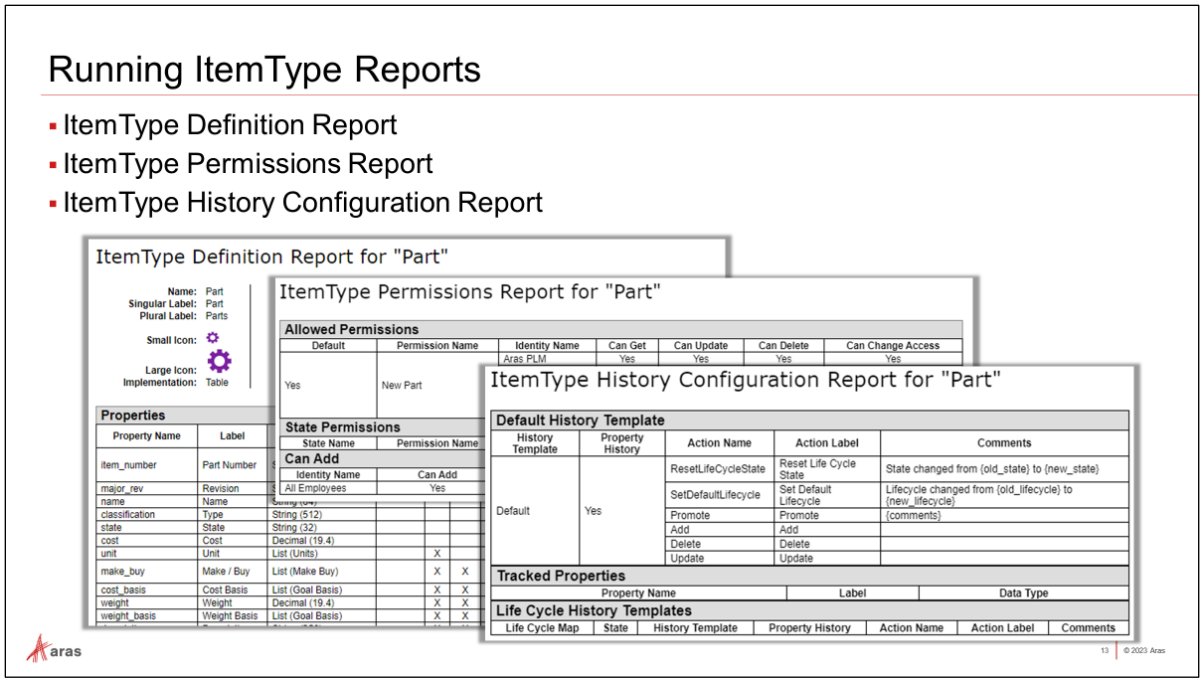
Each of these subclasses can have their own:

- Additional specific properties (in addition to the properties defined by the parent ItemType)
- Lifecycle Map
- User Interface (View of a form)
- Can Add capability

The following choices are available for the field named Show Parameters Tab:

- **Never** – never show the Parameters tab on Items of this ItemType.
- **When Populated** – only show the Parameters tab on Items of this ItemType when classified properties are available.
- **Always** – always show the Parameters tab on Items of this ItemType.

The Show Parameters Tab only becomes active in the special case when using a classification and would show a proper tab in the Relationship accordion.



Running Item Type Reports

There are three useful reports which summarize the various ItemType characterizations. Examples of each are shown above. Reports related to the ItemType can be run from the Reports button on the Toolbar of an open ItemType Template or from the context menu of the ItemType Search grid. These reports can be printed or saved using standard browser options.

ItemType Definition Report

The ItemType Definition Report summarizes the contents of the ItemType settings, all the custom Properties and system properties that have been changed, and the contents of all relationship tabs. It is a very fast and convenient way to see an ItemType's content without having to open the ItemType.

ItemType Permissions Report

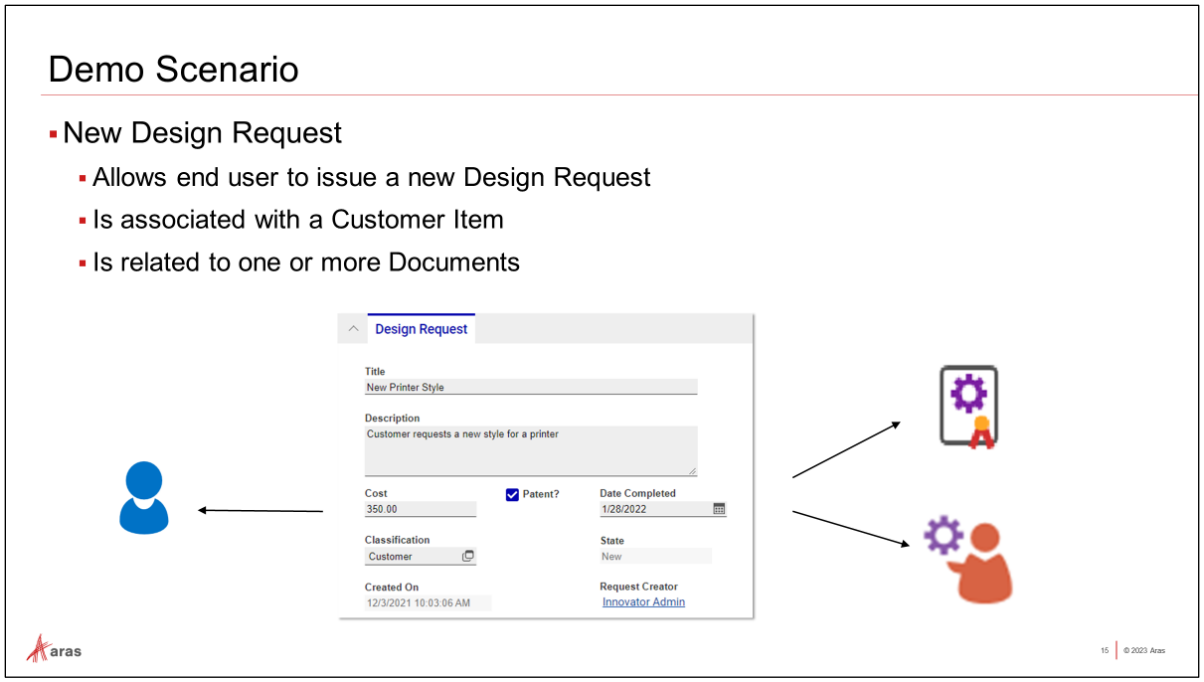
The ItemType Permissions Report summarizes the allowed identity permissions (Security) showing which identities Can Get, Can Update, Can Delete, and Can Change Access. It also shows which identities can create new items of this type under the Can Add settings. If there were specific permission changes, based upon a Lifecycle State change, they would appear in the State Permissions section.

ItemType History Configuration Report

The ItemType History Configuration Report summarizes the Default History Template, any Tracked Properties, and Lifecycle States which modify the History setting. If no History Template is assigned, the Report is displayed indicating that "This ItemType does not have a History Template defined."

Try it … Run an ItemType Report

1. On the tab containing the ItemType Part template, click the Reports button on the toolbar to see the three ItemType Report choices.
2. Click on each of the Reports to run them.



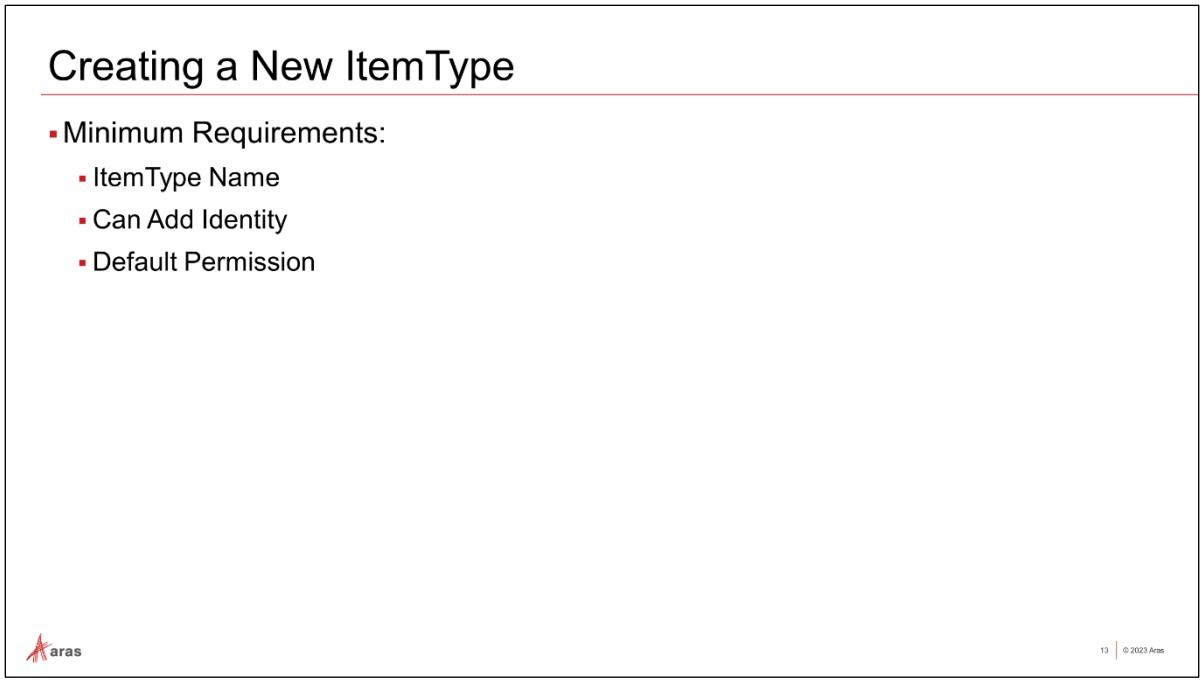
Demo Scenario

For demonstration in class, we will be working with a simple example scenario to illustrate various features in Aras Innovator.

A large company that sells consumer electronics would like to track Design Requests made by both Customers as well as internal employees of the company. Submitted Design Requests are then reviewed by appropriate staff members to either approve or reject the request before changes to the product line are made.

The Design Request should be linked to supporting documents as well as the customer that made the request.

Can you identify the ItemTypes we will need to accomplish this solution?



Creating a New ItemType

For a user to be able to create an Item of any ItemType, some minimum requirements must be met.

ItemType Name

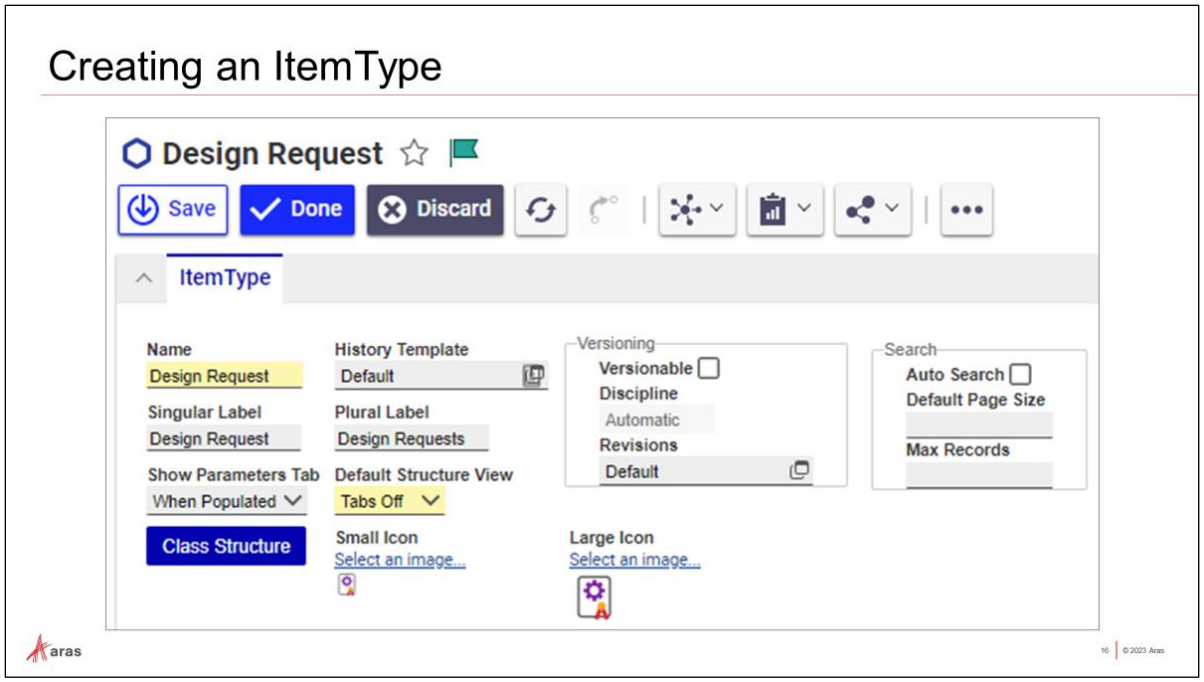
The ItemType Name identifies the type of Item to the user (e.g., Part, Document, etc.). The system also creates a unique internal id for the Item in the database to ensure exclusivity.

Can Add Identity

You need to indicate what Users in the system can create a new Item of this type in the database. This is typically assigned to a group Identity.

Default Permission

You will need to assign a default Permission to the ItemType to indicate what access one or more users have to the new Item.



Creating an ItemType

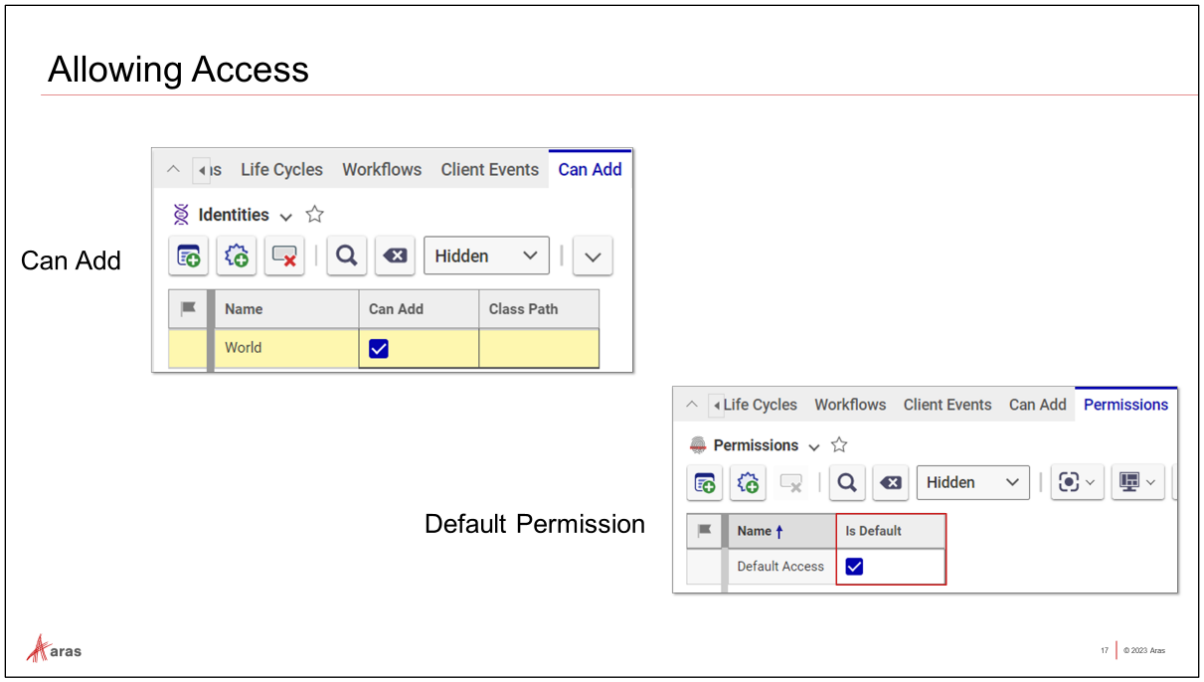
To support the suggested demo scenario, a new ItemType must be created.

Try it … Create the Design Request ItemType

1. Navigate to Administration > ItemTypes and create a new ItemType.
2. Enter or select the following values on the ItemType Form:

Name	Design Request
Singular Label	Design Request
Plural Label	Design Requests
History Template	Default
Default Structure View	Tabs Off
Auto Search	Yes [selected]
Small/Large Icons	Choose from provided image list

3. Save the Design Request ItemType. You will see 23 System Properties added automatically to the Properties tab.



Allowing Access

Administrators must define who is allowed to create new items by granting them Can Add privileges. They also identify on the Permissions tab a default Permission that will be assigned to each new item that is created. Depending on access permissions, the new ItemType creates a category that is placed in the TOC and can also be displayed on the main window's tile view.

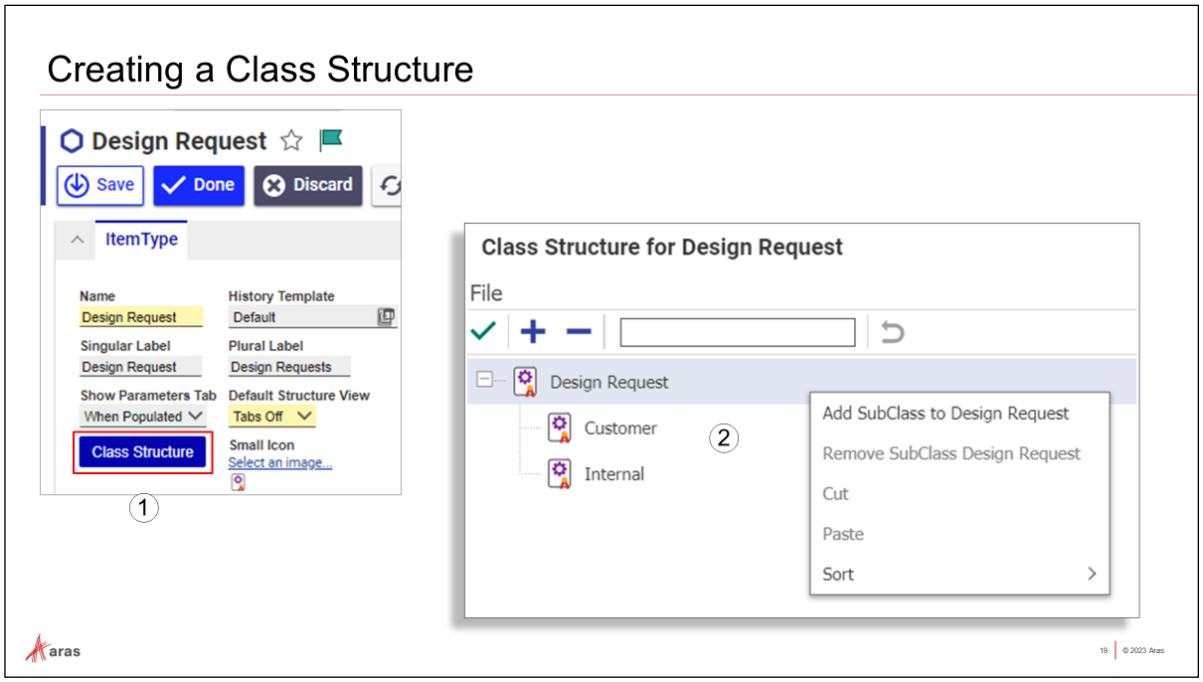
Try it … Provide Access Permissions to the Design Request ItemType

1. Provide the following values by clicking on each Relationship Tab and selecting the desired item (the Add button):

Relationship Tab	Name Value
Can Add	World (Can Add is checked)
Permissions	Default Access (check Is Default)

   Note: for initial testing, we allow everyone in the system (World) to add new Design Requests. This can be changed later on.

2. Save the ItemType.



Creating a Class Structure

A class structure can be added to any ItemTypes to distinguish between different types of these items. Properties can be added to the ItemType and assigned to a specific Classification (Class Path). Forms can then be created to show different views based on the selected Classification.

Classes can be alphabetically sorted once created by clicking on the top level class and using context menu to sort ascending or descending.

Note: classification, class structure, and class path are synonymous.

Try it … Create a Class Structure

For the Design Request ItemType we will create two subclasses named: Internal and Customer.

1. Edit the **Design Request** ItemType and click the *Class Structure* button to launch the Class Structure window.
2. Select the Design Request top-level class, right-click and select Add Subclass to Design Request.
3. Enter *Internal* in the field provided and Tab or Enter to accept the value.
4. To create a second subclass at the same level, select the Design Request top-level class once more, right-click and select Add Subclass to Design Request.
5. Enter *Customer* in the field provided.
6. Click the Save and Close button (green checkmark) to return to the ItemType.
7. For the Show Parameters Tab field, ensure the **Never** dropdown menu option is selected.
8. Click the Done button.
9. Run the ItemType definition report, then close the tab when done viewing the report.

Summary

In this unit, you learned how ItemTypes are configured and how to build new ItemTypes.

You should now be able to:

- Define the Purpose of an ItemType
- Create a new ItemType
- Display ItemType Properties
- Run an ItemType Report

Review Questions

1. What is the difference between an Item and an ItemType?

2. What are some key requirements you should consider when creating an ItemType?

3. Which report best summarizes the contents of an ItemType?

4. An administrator creates an ItemType, but the end users cannot see it in the TOC Pane – why?

5. What are advantages of using an ItemType classification?

Lab Exercise Scenario

Part Change Request – Solution Requirements

Each unit of this course contains an additional exercise so you can practice what you learned.

As a working example, you build a basic product management system that allows a company to record and process part change requests made by customers about existing goods that they sell. You classify the requests as either defects or enhancements and define an appropriate Lifecycle map and Workflow Map for the solution.

The Part Change Request must be able to:

- Collect information about the part issue including a description, reason, and cost.
- Establish a required connection to a Customer Item and one or more Part Items.
- Log one or more review notes for tracking progress.
- Follow a prescribed Lifecycle map - and once closed force the request to remain read only.
- Support a workflow that issues assignments with specific tasks to various parties to either approve or reject the request as efficiently as possible.
- Include various classifications.

Each unit of the course allows you to add new functionality to support these requirements.

The first lab exercise follows.

Lab Exercise

Goal

Create a Change Request ItemType.

Scenario

Use Aras Innovator to create a basic change management system that allows request items to follow a prescribed Lifecycle map. In this lab, you first define and configure the Change Request ItemType so that users can create one or more Change Request Items.

Steps

1. Make sure you are logged on to the correct database as an Administrator (your student login).
2. Create an ItemType.
3. Provide the following information for the new Item Type:

Name	Change Request
Singular Label	Change Request
Plural Label	Change Requests
History Template	Default
Show Parameters Tab	Never
Default Structure View	Tabs Off
Search - Auto Search	Yes [Selected]
Search - Default Page Size	20
Search – Max Records	100

4. Select a small and large icon to represent the ItemType in the user interface.
5. Create a class structure to support these subclasses:

Enhancement
Defect

6. Save the Change Request ItemType.

   Next, you need to allow users to be able to add new Items of type Change Request to the database. For initial testing, we will allow everyone in the system (World) to add new Change Requests. Later in the course we will fine tune the security of this ItemType to a select set of users.
7. Configure the following for the Change Request ItemType using the Relationship tabs on the ItemType form:

Can Add	World
Permissions	Default Access – (Ensure this is set as the Default permission)

8. Click the Done button to Save and unclaim the Change Request ItemType.

   Note that, if you had been able to create a Change Request Item, the Tab title would show a simple naming pattern to describe the Item, usually the GUID.

Unit 4 The Table of Contents (TOC) Editor

Introduction

In this unit, you learn how to use the new Table of Contents (TOC) Editor.

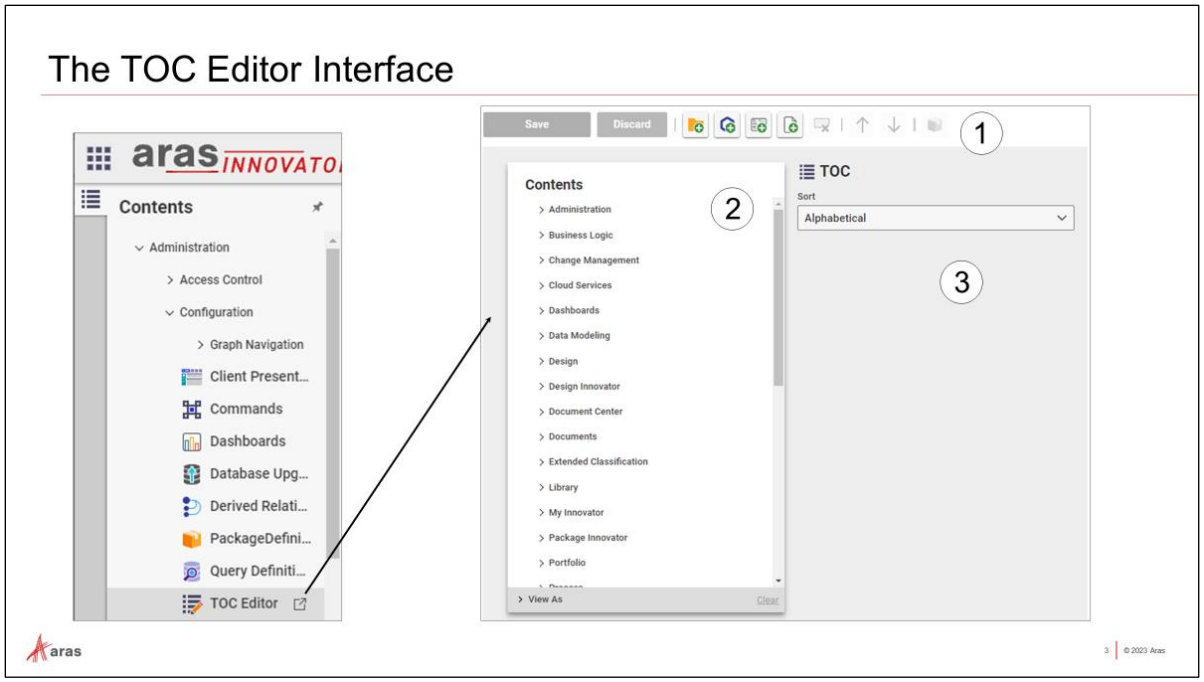
Course Goals

- Describe the benefits of the TOC Editor
- Describe the TOC Editor interface and Toolbar buttons
- Create and Move a Category
- Add an ItemType button to a Category
- Customize the ItemType button in the TOC
- Preview the TOC as a specific Identity

Introduction to the TOC Editor

The TOC Editor provides a graphical interface to view and configure the content of the TOC. The TOC Editor simplifies the process of editing the TOC with a single interface. Now, administrators can see and configure the entire TOC without opening each individual ItemType. The TOC Editor allows you to see all categories, including all nested categories and their ItemTypes. It allows you to create categories, pick icons for each category, assign labels to each one, associate ItemTypes with categories, move categories, and preview the TOC as a specific identity. The TOC Editor offers several benefits to Aras Innovator Administrators and developers:

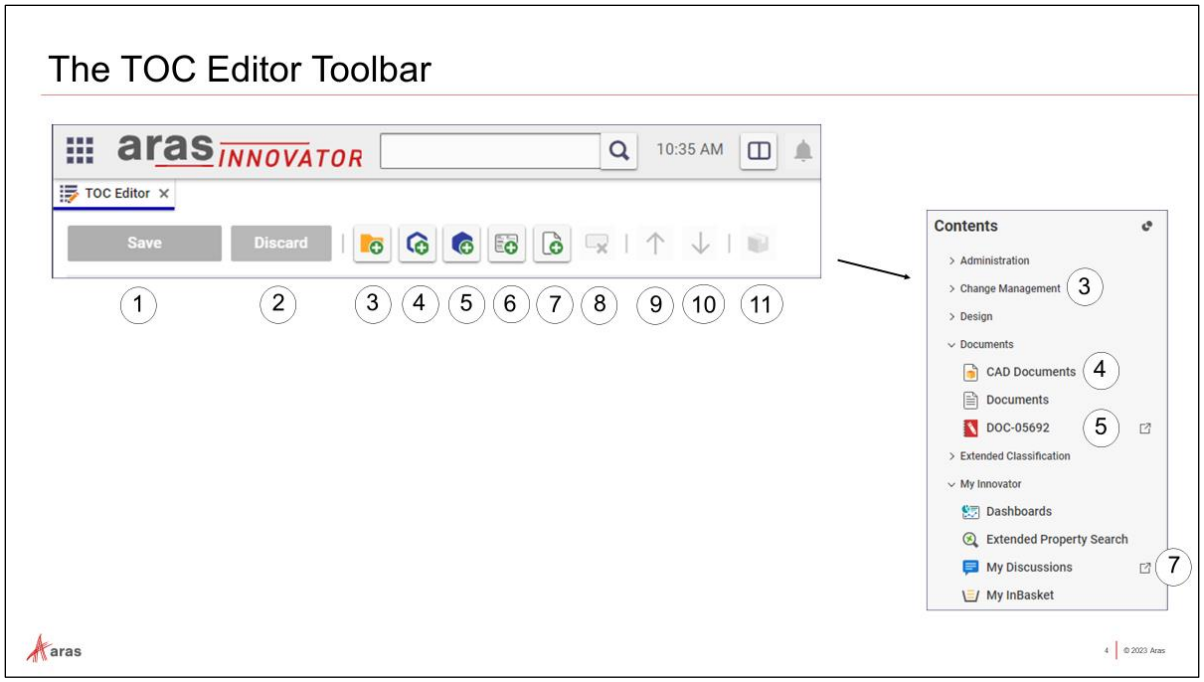
- **Reduced Context Switching:** Administrators can configure all TOC content in a single interface. No need to switch between tabs and edit individual ItemTypes. This is a change from previously using the TOC Access relationship tab on each ItemType.
- **Improved Visibility:** Administrators can instantly see the results of their edits without logging out and logging back into Aras Innovator. The "View As" functionality enables administrators to see the TOC view for different users and groups, making it easier to test and troubleshoot their updates.
- **Simplified Packaging:** The TOC Editor automatically packages the CUI objects that are used to render the TOC controls.



The TOC Editor Interface

The TOC Editor is accessed from the Administration > Configuration category of the TOC. It is comprised of three main sections: the Toolbar, the Contents pane, and the Details pane.

①	The TOC Editor Toolbar	Contains the controls for editing the TOC.
②	The Contents pane	Displays all Categories and ItemType buttons that are configured in the Aras Innovator database.
③	The Details pane	Displays a form for viewing and editing the properties of the selected item or category in the Contents Pane.

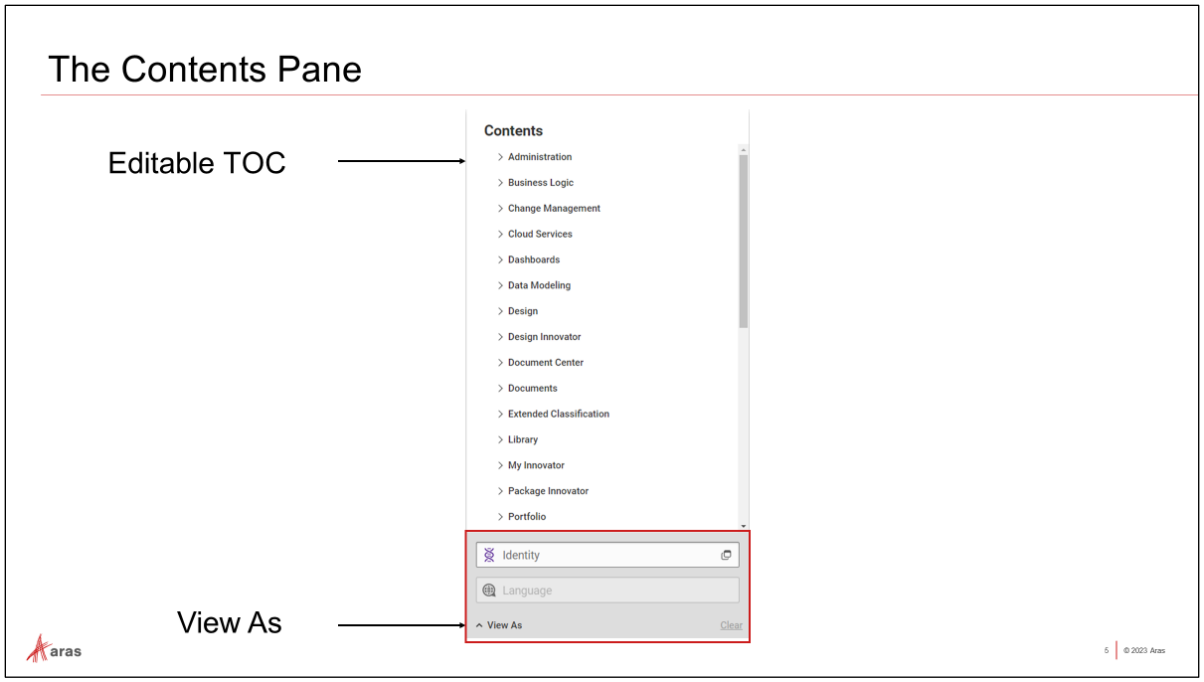


The TOC Editor Toolbar

The TOC Editor toolbar contains the following controls:

①	**Save**	Saves the current changes in the editor. A warning appears about unsaved changes if the tab is exited before Save. This button is disabled if there are no changes in the editor since the previous Save.
②	**Discard**	Discards the current changes in the editor.
③	**Add Category**	Adds a new Category to the TOC. If another Category is selected, the new Category will be created as a sub-category of the selected Category. If no Category is selected, the new Category will be located at the bottom of the TOC.
④	**Add Item Type**	Adds a new ItemType button to the TOC. If a Category is selected, the new ItemType button is created as a child of the selected Category. If no Category is selected, the new ItemType button is located at the bottom of the TOC.
⑤	**Add Item**	Places a single item in the TOC.
⑥	**Add Form**	Allows an administrator to configure a form to display when a TOC button is clicked.
⑦	**Add Page**	Allows an administrator to configure an HTML page to display when a TOC button is clicked.
⑧	**Delete**	Deletes the currently selected Category or ItemType. If a Category includes ItemTypes, and the Category is deleted, the ItemType buttons remain in the TOC.
⑨	**Move Up**	Moves the selected Category or ItemType button within its parent Category. This button is disabled if the selected Category or ItemType button cannot be moved higher.

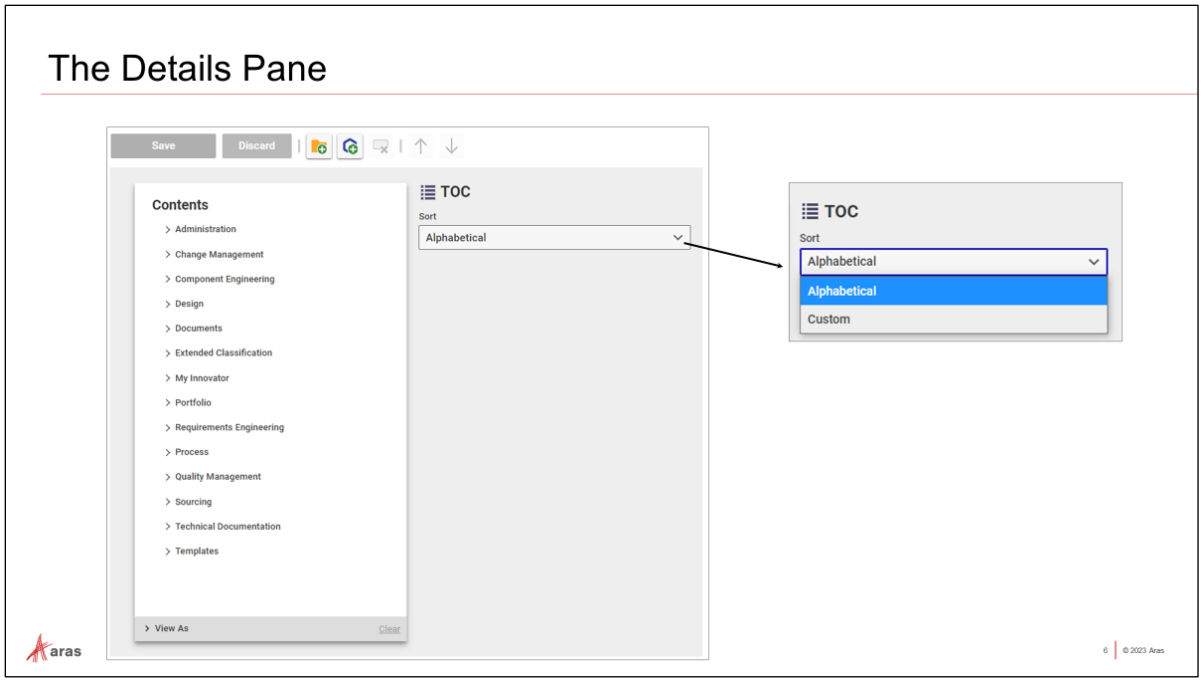
⑩	**Move Down**	Moves the selected Category or ItemType button within its parent Category. This button is disabled if the selected Category or ItemType button cannot be moved lower.
⑪	**Add to Package Definition**	Add a newly created category or dashboard item to a Package Definition



The Contents Pane

The pane on the left side of the TOC Editor displays all Categories and ItemType buttons that are configured in the Aras Innovator database.

Use the View As filters at the bottom of the Contents Pane to preview the TOC configuration for a specified user, group identity, or language. The TOC Editor enters a read-only mode when the View As filter is active. Use Clear to return to Edit mode.

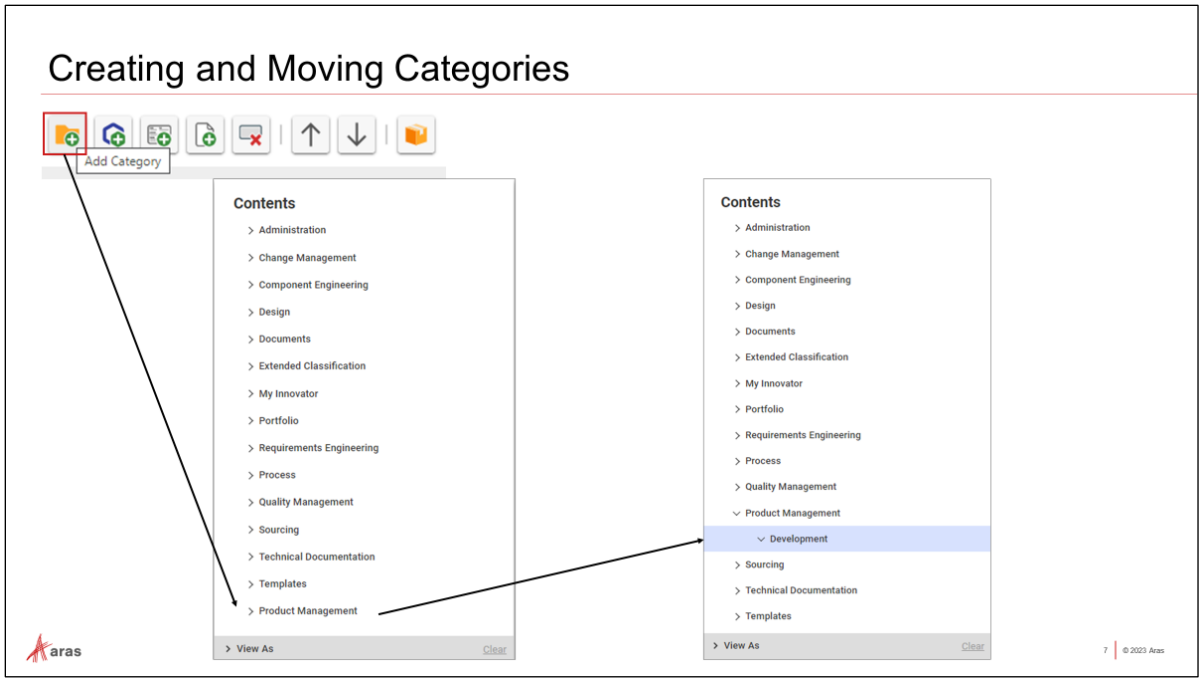


The Details Pane

When no items are selected in the Contents Pane, the Details Pane shows general properties for the TOC. This is the default view when the TOC Editor opens. To deselect an item in the Contents Pane and show the TOC properties, click the Contents header in the Contents Pane.

Sort determines the order that the Aras Innovator web client renders the TOC elements. The Contents Pane always displays the Custom Sort view if one has been created. Therefore, the actual TOC may appear different from the Contents pane when the Sort property is set to Alphabetical.

- **Alphabetical:** The default setting is Alphabetical, which sorts TOC Categories and their child items in alphabetical order. This setting is consistent with the behavior of Aras Innovator 12 and earlier releases.
- **Custom:** This setting sorts the TOC Categories and ItemTypes according to the order defined in the TOC Editor.



Creating and Moving Categories

To create a parent category at the same level, consistent with other categories in TOC, deselect any categories by clicking on Contents. The new Category is created at the bottom of the TOC.

However, if an existing Category is selected when the Add Category button is clicked, the new Category is created as a sub-category of the selected Category. To move a category, either click on it and drag it into position or click on it and use the Move Up and Move Down buttons to move it.

Try it … Create a Category

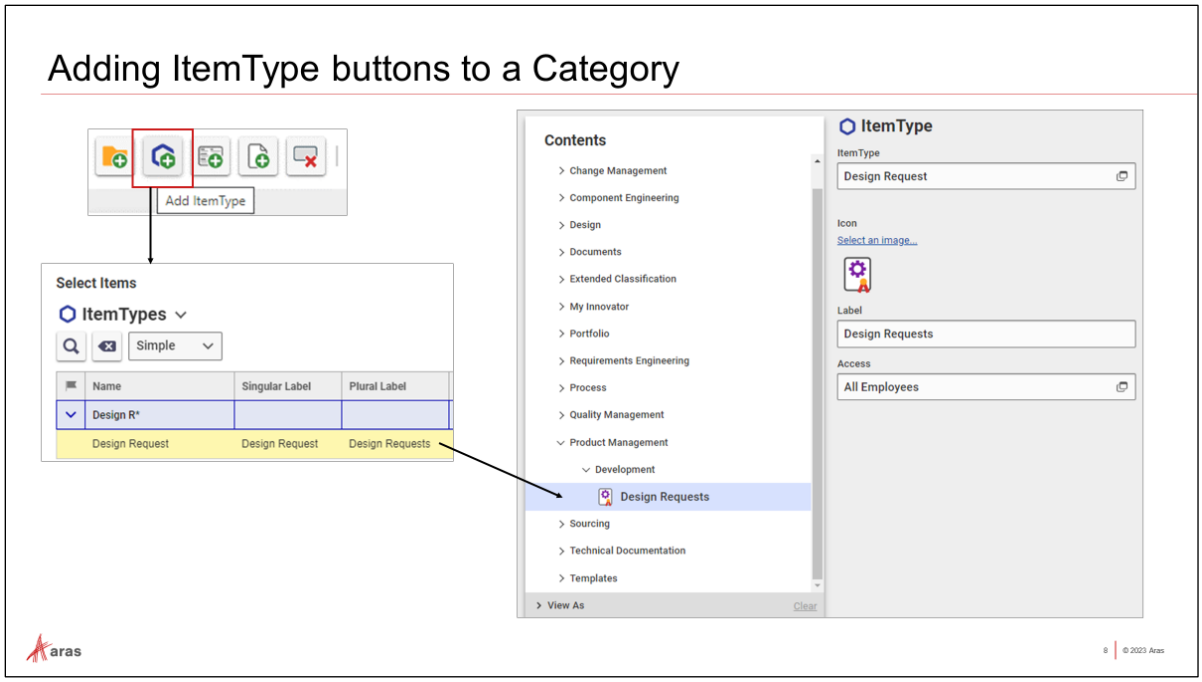
5. Navigate to TOC > Administration > Configuration and click on the TOC Editor.
6. Do not select an existing category. If you already clicked on a category, click on Contents to remove the category selection.
7. Click the Add Category button and enter *Product Management* for the Label.
8. Click Save.

Try it … Move a Category

1. Move the Product Management Category to directly above the Sourcing Category.
2. Click Save.

Try it … Create a Child Category

1. Click on the Product Management Category and click the Add Category button.
2. In the Label field, enter *Development* and click Save.



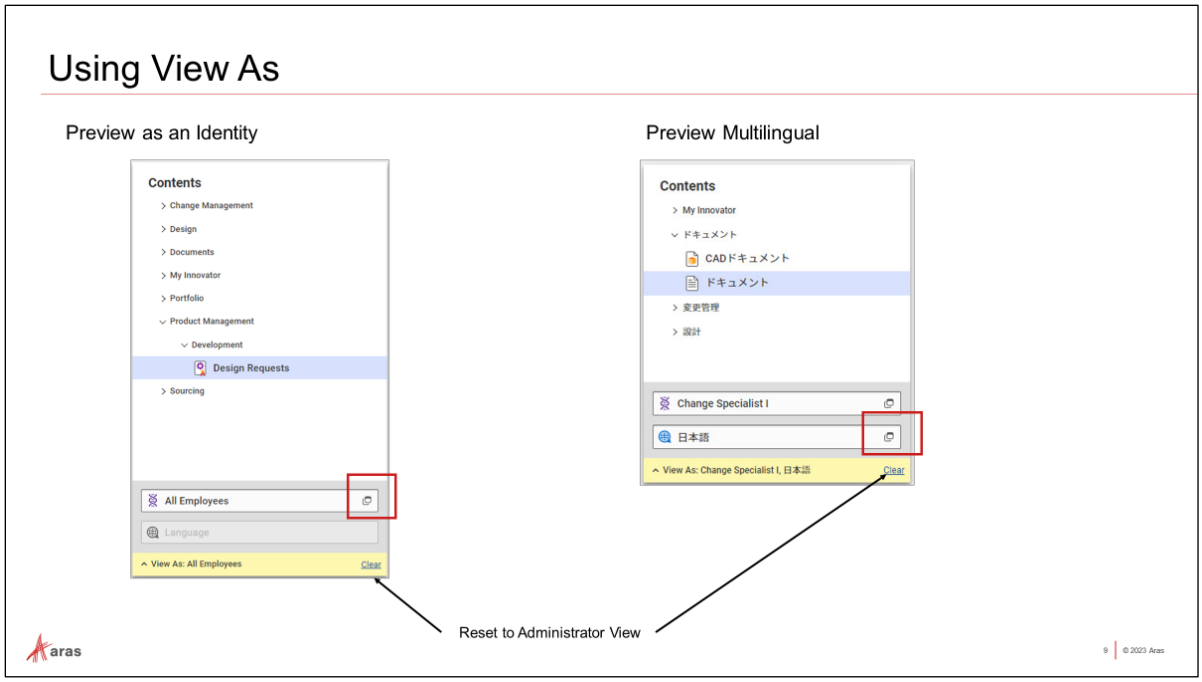
Adding ItemType buttons to a Category

The Add ItemType button adds a new ItemType button to the TOC. If a Category is selected when this button is clicked, the new ItemType button is created as a child of the selected Category. If no Category is selected, the new ItemType button is created at the bottom of the TOC.

Once an ItemType is added to the TOC, the default behavior of creating a new item or launching a search grid is enabled for this ItemType.

Try it … Add an ItemType button to a Category

1. With the newly created **Development** category selected, click the Add ItemType button.
2. Search for and select the *Design Request* ItemType and click OK.
3. Accept the default Label value and for the Access field, enter *All Employees* or search for and select *All Employees*.
4. Click Save.

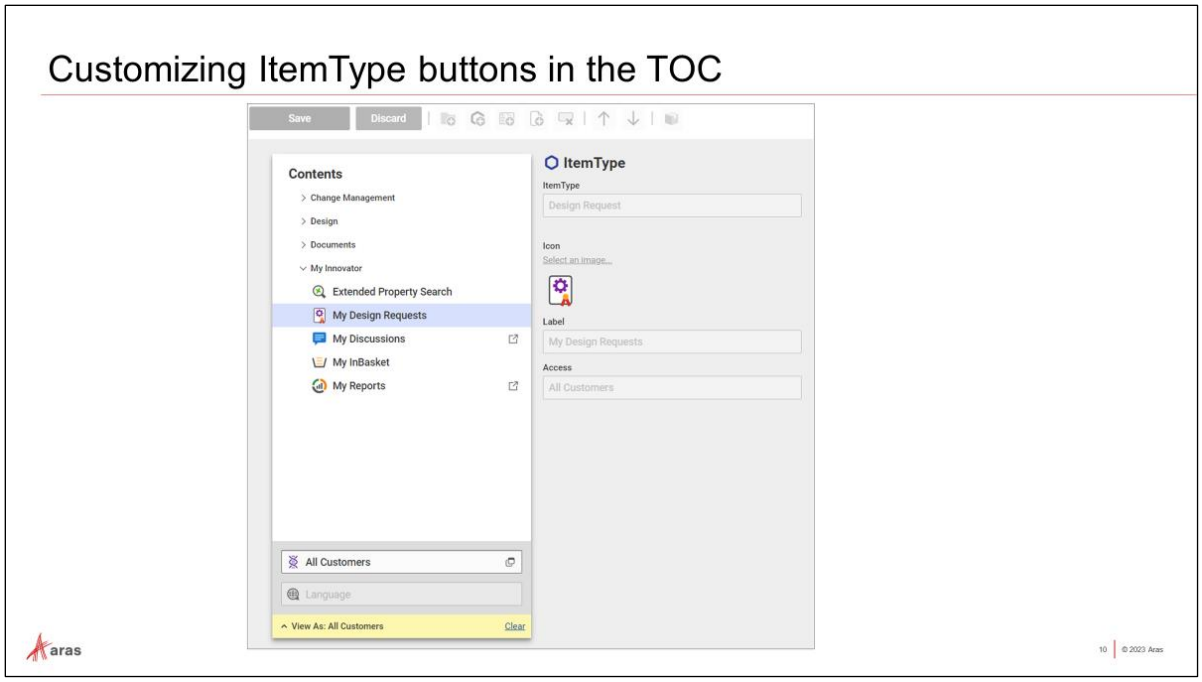


Using View As

The TOC Editor's View As feature allows you to quickly see how changes to the TOC impact different Identities without having to log out and back in. The View As filters also allow an administrator to preview the TOC configuration in another language. After the admin chooses a language configured in the database, the TOC Editor content will refresh and display the multilingual values stored for that language. The TOC Editor enters a read-only mode when the View As filter is active.

Try it … View the TOC using the All Employees Identity

1. In the Contents pane, click View As.
2. For the Identity field, click the Select items button.
3. In the Select Items window, search for and select the **All Employees** Identity.
4. Click Clear to return to Edit mode.



Customizing ItemType buttons in the TOC

The Details pane displays a form for viewing and editing the properties of the selected row in the Contents Pane. When no items are selected in the Contents Pane, the Details Pane shows general properties for the TOC. This is the default view when the TOC Editor opens. With an ItemType button selected in the Contents pane, the following items can be edited:

- **ItemType:** Sets the ItemType associated with the button. When the user selects the button in the TOC, the Navigation Pane will display the user's favorite searches, favorite items, and common actions for that ItemType.
- **Icon:** Defines the icon that appears on the button in the TOC.
- **Label:** This multilingual property defines the text that is shown on the button in the TOC.
- **Access:** Defines which identities see this button in the TOC.

Note that each button can only be configured for a single Identity. To display an ItemType in the TOC for multiple Identities, either create a separate button for each Identity or use a group Identity that encompasses all individuals or groups who should see that ItemType in the TOC.

Try it … Customize an ItemType button in the TOC

We want to allow customers to create and view their Design Requests from the My Innovator Category.

1. In the TOC Editor, select the My Innovator Category and using the Add ItemType button, add the **Design Request** ItemType.
2. Change the Label to *My Design Requests*.
3. Change the Access field's value to *All Customers*. If desired, change the icon.
4. Save the changes.
5. Use View As **All Customers**. Verify that all customers can see 'My Design Requests' under 'My Innovator' category, and cannot see 'Product management' category.
6. Click Clear to return the TOC Editor to Edit mode.

Summary

In this unit, you learned how to use the new TOC Editor.

You should now be able to:

- Describe the benefits of the TOC Editor
- Describe the TOC Editor interface and Toolbar buttons
- Create and Move a Category
- Add an ItemType button to a Category
- Customize the ItemType button in the TOC
- View the TOC as a specific Identity

Review Questions

1. What can you add to the TOC using the TOC Editor?

2. How would you get a preview of the TOC without logging in as a different user?

Lab Exercise

Goal

Add the Change Request to the Table of Contents.

Scenario

Use the TOC Editor to add the Change Request ItemType to the Product Management category.

Preparation

If you did not create the Product Management category earlier with the instructor, create it now.

1. Navigate to TOC > Administration > Configuration and click on the TOC Editor.
2. Do not select an existing Category.
3. Click the Add Category button and enter *Product Management* for the Label.
4. Click Save. The new **Product Management** category is created at the bottom of the TOC.
5. Move the **Product Management** category directly above the Sourcing category.
6. Click Save.

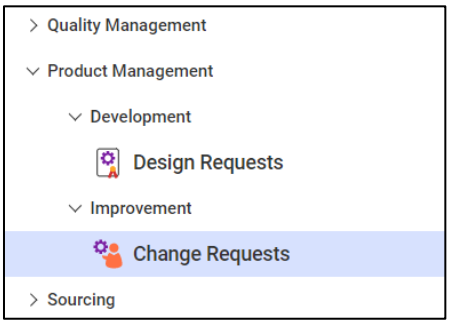
Steps

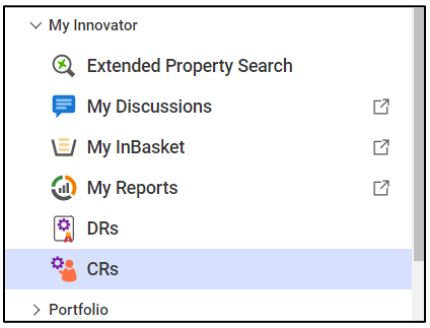
Create a Child Category for Change Requests

1. Under the Product Management category, add a child category named *Improvement*.
2. Add a **Change Request** ItemType button to the **Improvement** category.
3. Change the Access to *All Employees* and save.

Customize an ItemType button in the TOC

1. Use the TOC Editor to add the **Change Request** ItemType to the My Innovator category.
2. Change the Label to *CRs*.
3. Change the Access field's value to *Engineering*.
4. If desired, change the icon.
5. Save the changes.
6. Use View As **Engineering**.
7. Click Clear to return the TOC Editor to Edit mode.





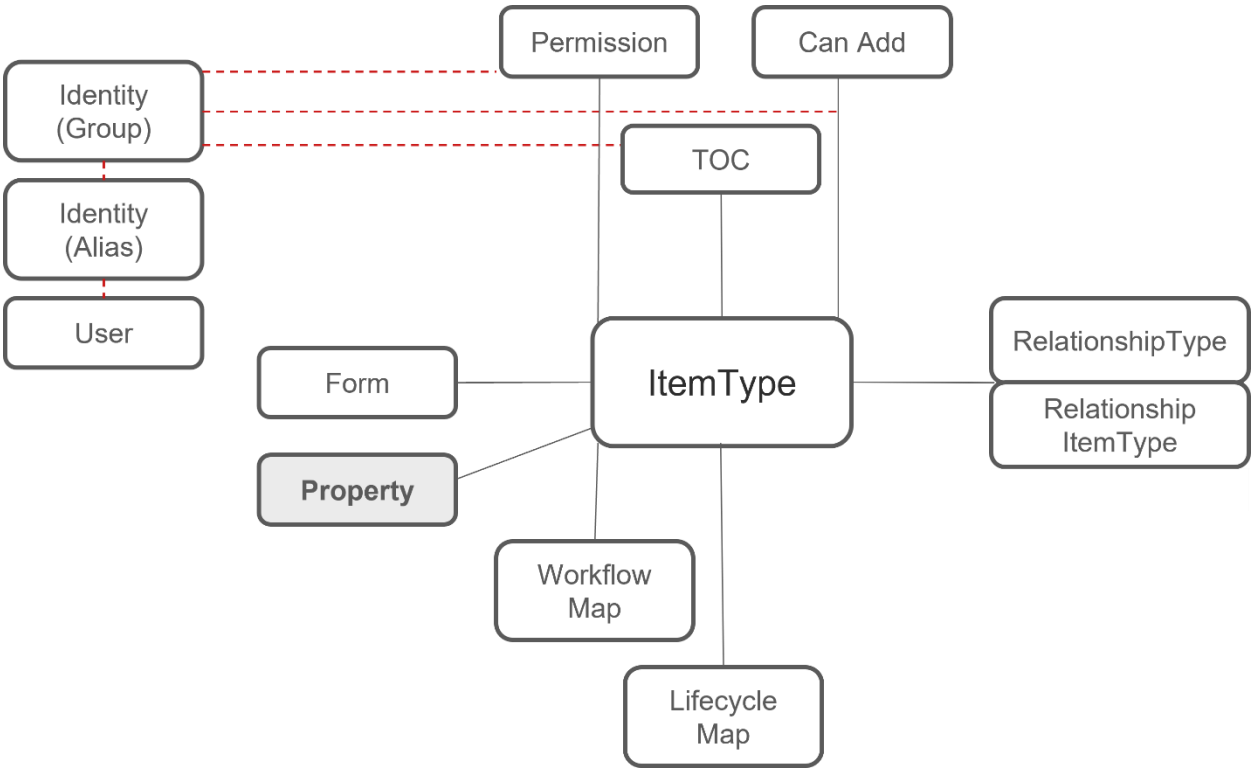
Unit 5 Itemtype Properties

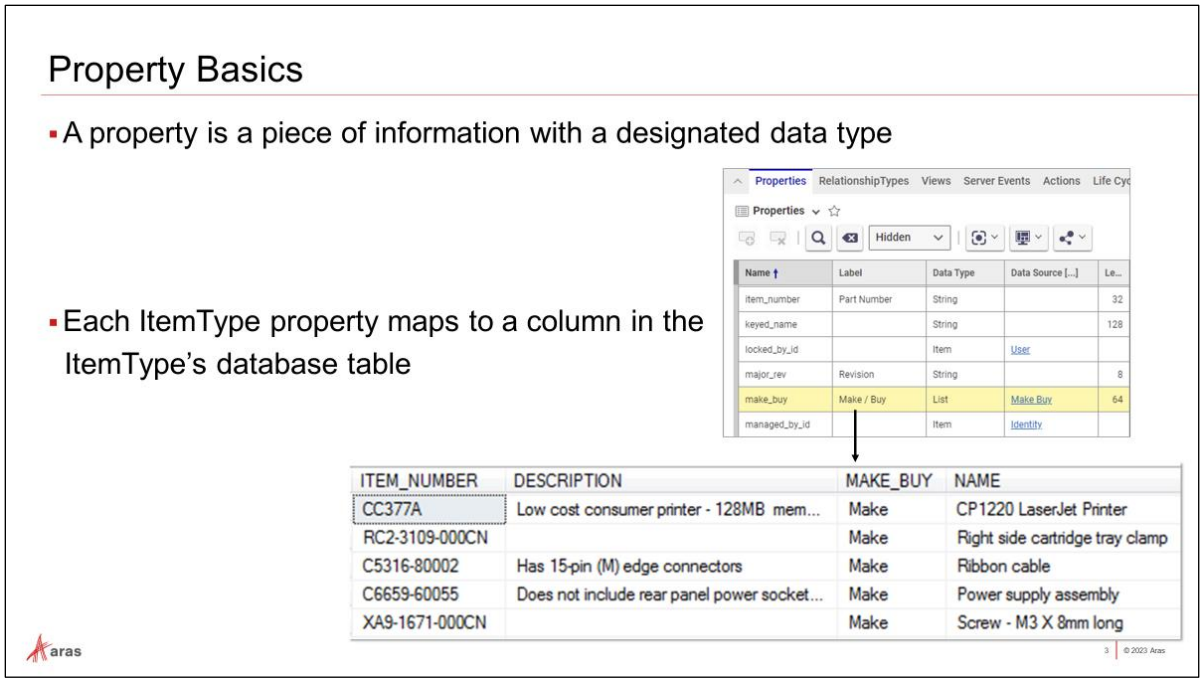
Overview

In this unit, you learn how to assign properties to an ItemType. You also learn how properties are used in the database.

Objectives

- Describe Property Basics
- Identify Property Types
- Explore Property Data Types and Settings
- Define an ItemType Property





Property Basics

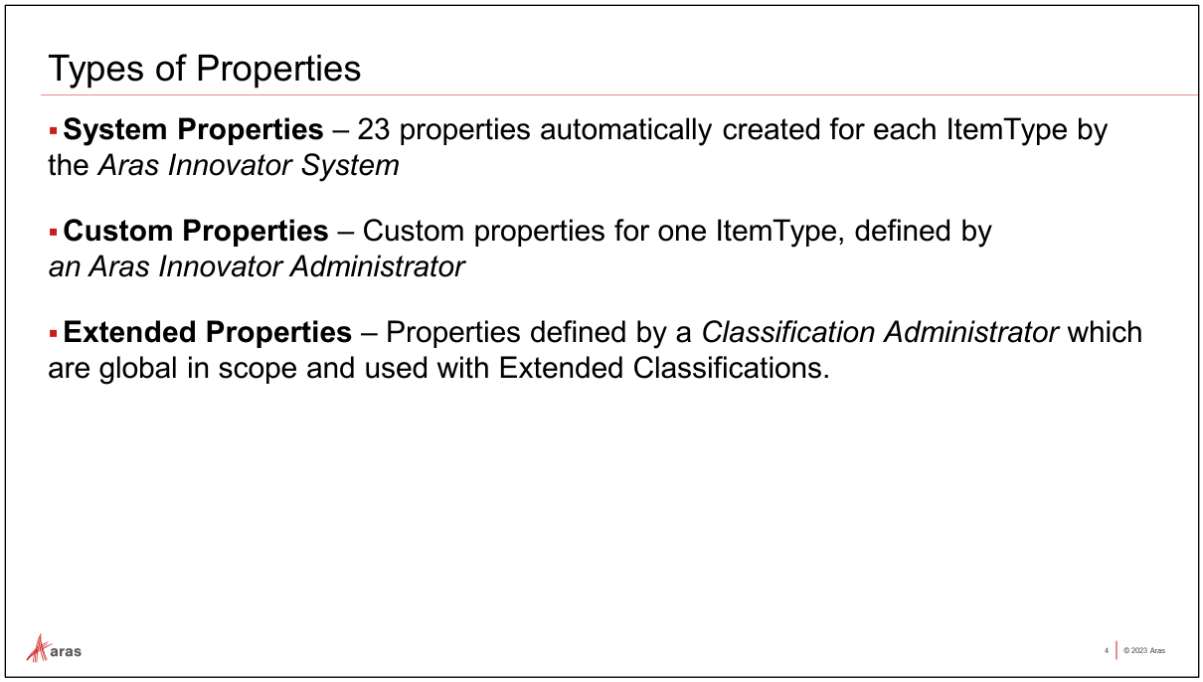
The Properties tab on the ItemType form contains all properties for a given ItemType (both system properties and custom properties). Property names are saved and stored using lowercase characters and underscores replace any spaces.

ItemType Properties support a variety of Data Types, which are handled automatically by the Client and Server for presentation and storage purposes. Properties can optionally be initialized with default values, configured to be required (not null), enforce uniqueness, or formatted.

Property settings can be configured to arrange search grids include sorting, column order, labels (spaces allowed), display widths, and whether they are visible or hidden completely. Properties also correspond directly to Fields in User Interface Forms, which will recognize a Property by Data Type and provide the correct behavior.

The ItemType schema is represented in its corresponding database table, which contains a column for each property from the Properties tab. Specifically, the table contains:

- A row for each Item instance of the ItemType (Item instance)
- A column for each Property under the ItemType
- Values in each Property column for each Item instance row



Types of Properties

Properties are also referred to as attributes or meta data. Aras Innovator supports properties that are set by the server as well as custom properties created by administrators.

System Properties

System properties are created by the Aras Innovator System when a new ItemType template is created. Some of the included properties are: creation date, created by, Lifecycle State, revision, generation, etc. System Properties are used by the functionality of Aras Innovator; therefore, they should not be deleted, and their functionality must not be modified.

Custom Properties

Custom properties are created by an Aras Innovator administrator for a specific ItemType to store custom information specific to the ItemType. For example: Title, Description, Completion Date, Patent, etc.

Extended Properties

Extended Properties are a pool of properties created on a global scope by a Classification Administrator. They are often used to store information that is common across ItemTypes, such as model, series, dimensions, color, weight, shipping method, etc. Extended Properties can be organized into a hierarchical classification tree (Extended Classification). Alternatively, they can be maintained programmatically.

System Properties

System properties are created with every ItemType and must be present for the ItemType to function correctly

- Each Item contains an id which is a 32 character unique identifier (GUID)
- A config_id is also maintained which links multiple versions of the same Item

	ITEM_NUMBER	NAME	ID	CONFIG_ID	GENERATION
1	C6229A	4MB SIMM memory module	884BC6BBA2A4441387DEA55BE669E5D1	884BC6BBA2A4441387DEA55BE669E5D1	1
2	C6230A	8MB SIMM memory module	7DD57AC66AC2410E906AFB7F4025CA6D	7DD57AC66AC2410E906AFB7F4025CA6D	1
3	C6231A	16MB SIMM memory module	A77BA990EEA549A08DA7538BB5451FD6	A77BA990EEA549A08DA7538BB5451FD6	1
4	C6231A	16MB SIMM memory module	CC9E7233BE7A442BB224F42A2CBC5F3B	A77BA990EEA549A08DA7538BB5451FD6	2
5	C6231A	16MB SIMM memory module	1DA899BADDB0453A89E79E4CFD2C6C44	A77BA990EEA549A08DA7538BB5451FD6	3

aras

5 | © 2023 Aras

System Properties

When an ItemType is saved, the system creates 23 System Properties by default, as follows:

Property	Description	Property	Description
classification	class path of the item	major_rev	major revision of versioned item
config_id	common id across generations of an item	managed_by_id	id of Manager identity
created_by_id	id of user that created the item	minor_rev	not used
created_on	date item was created	modified_by_id	id of last user that changed item
css	cascading style sheet	modified_on	date item changed
current_state	id of current Lifecycle State item	new_version	
generation	sequential version number	not_lockable	true (1) if item cannot be claimed (i.e., edited)
id	a 32 character globally unique identifier (GUID) for the item	owned_by_id	id of Owner identity
is_current	true (1) if this is the most current version	permission_id	id of the current Permission item
is_released	true (1) if this item is released	state	current state (text)
keyed_name	value of the keyed name, in order	team_id	id of current Team item
locked_by_id	id of user that has claimed item		

Custom Properties

- Custom properties are created by an Aras Innovator Administrator
- They are unique for each ItemType
- They are used to store values for individual items for example:
  - Name
  - Number
  - Description
  - Location
  - Patent

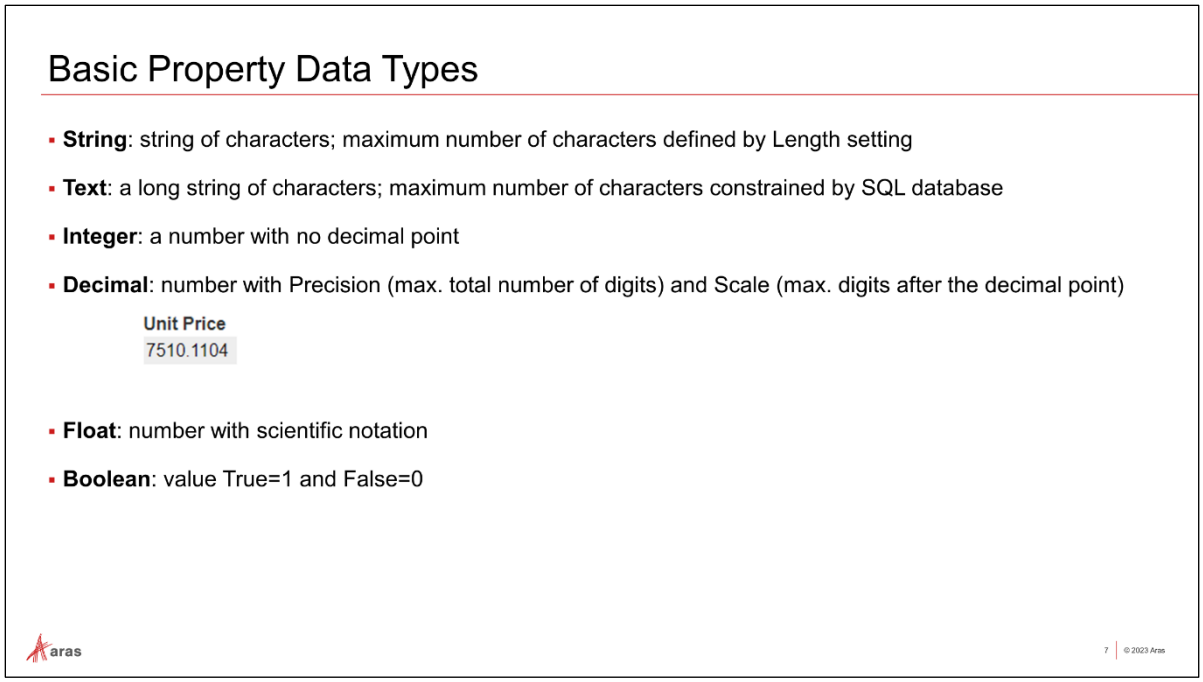
aras

6 | © 2023 Aras

Custom Properties

Custom properties are created by an Aras Innovator administrator for a given ItemType to store information specific to that ItemType according to the specified Data Type.

In the training, the custom properties will start with an underscore, to ease the distinction with system properties.

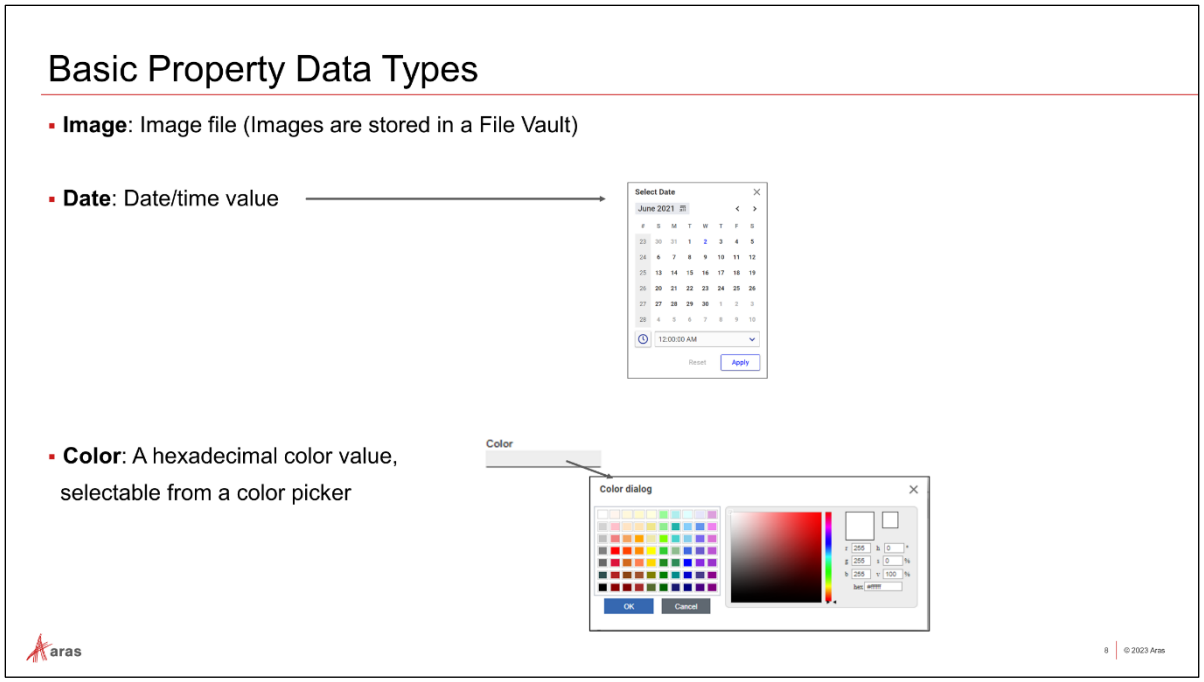


Basic Property Data Types

The User Interface will recognize each Data Type setting automatically for input/output purposes. The underlying Database also stores Property values according to the configured datatype.

Alphanumeric properties

A String has a fixed size (nchar). A Text has a variable string size (nvarchar), and the limit of characters is not defined by the system configuration but by the database, see nvarchar(max) on the Microsoft SQL documentation.

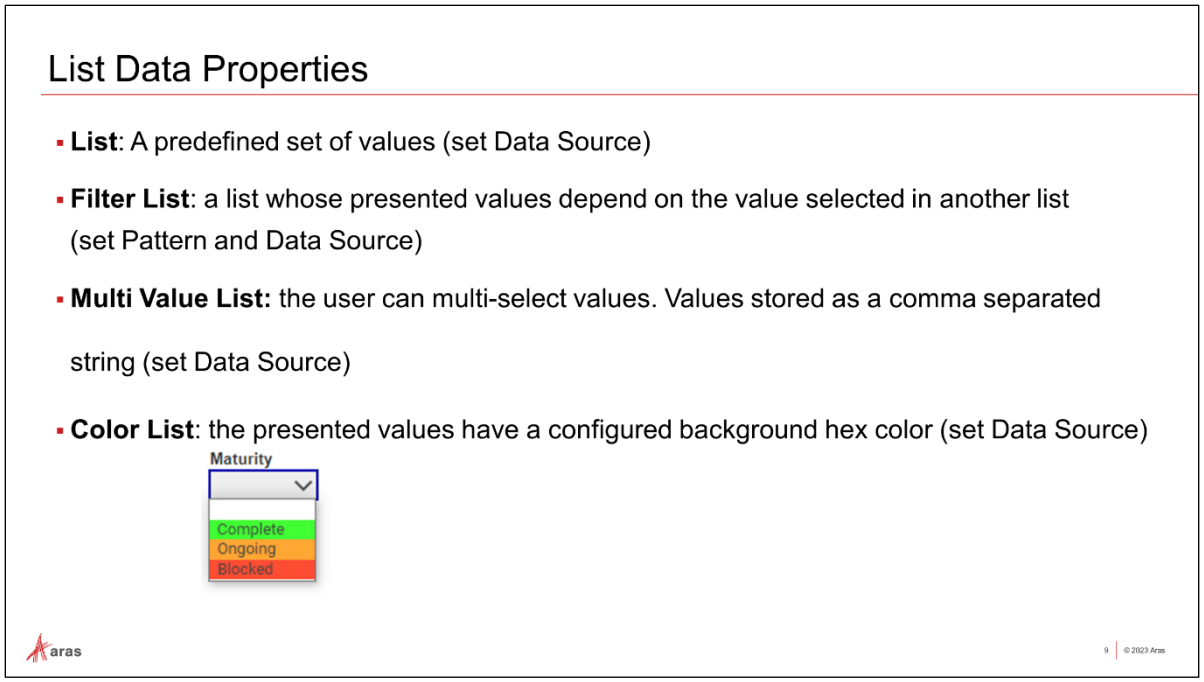


Basic Property Data Types

Several data types offer custom dialogs to make data entry easier. For example, the Date data type displays a calendar dialog to the user.

Images

Images are stored in a File Vault, while the other properties are stored in the database.



List Data Properties

Allows a user to select from a predefined list of options.

List

A List presents a list of choices that can display as a dropdown list, a radio button list or checkbox list to a user in a Form. The value displayed to the user is the label, and a corresponding property value is set in the database when the user makes a selection.

Filter List

A Filter List allows you to configure a cascading set of dropdown lists where the value selected in one list affects the values presented in a subsequent list.

Multi Value List

A Multi value List is the same as a standard List described above, except that the user can select one or more choices from the list. Selected choices are saved as a comma delimited string.

Color List

A dropdown list that displays a text label with a different color background for each row.

Note: the values in a Color List are the hexadecimal color number, therefore the Color List property value visible in the SQL database for Items will be a color number.

Special Property Data Types

- **MD5**: encrypted value. A cryptographic function typically used for passwords
- **Sequence**: auto-numbering pattern (Must set a Sequence Data Source)
- **Item**: a link to another item (set Data Source)
- **Foreign**: property value from a related item, shown as read-only (set Data Source)
- **Formatted Text**: long HTML formatted text value
- **Multilingual String:** text stored in multiple languages. Displayed text depends on Locale setting.

aras

10 | © 2023 Aras

Special Property Data Types

MD5

User passwords use this hash function, which produces a 128-bit hash value (and displays as a 32-digit hexadecimal number in the database).

Sequence

Use a Sequence for auto-numbering Items. A formatted pattern is set to increment a number value each time a new Item is created.

Item

The property value is an actual related Item (e.g. the Manufacturer related to a Manufacturer Part).

Foreign

The Foreign data type allows access to foreign properties that reside on another selected Item.

Formatted Text

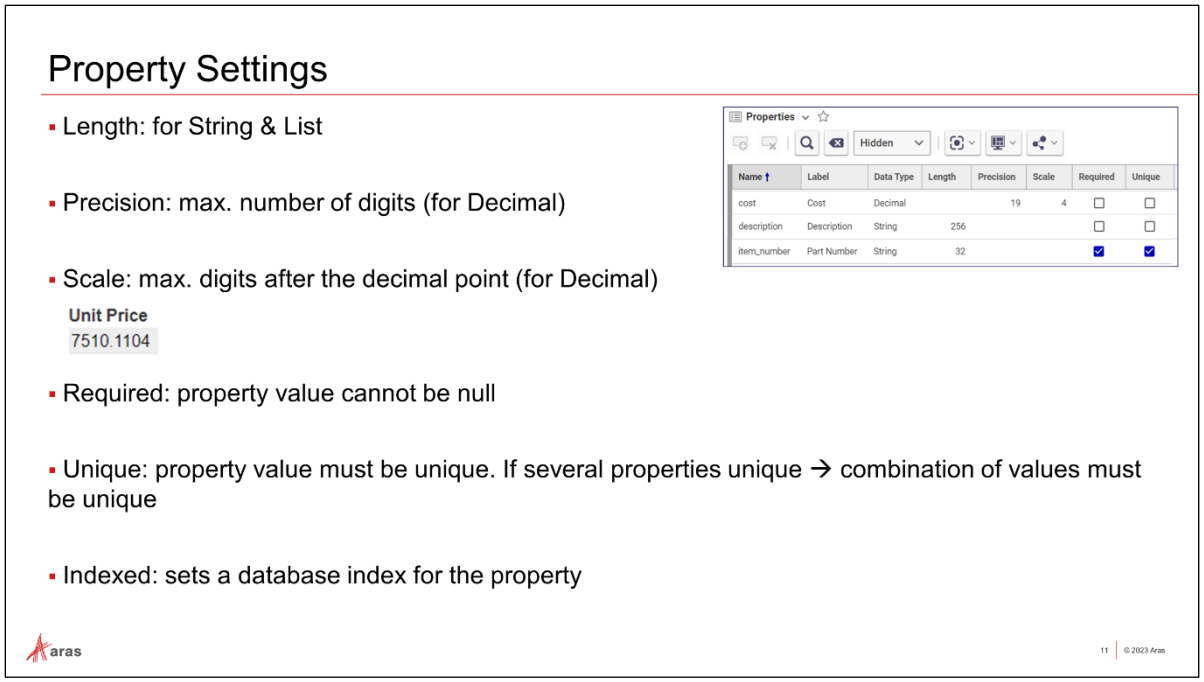
Text with formatting which can be edited with the "Innovator Formatted Text Editor" (e.g., bold text, italic, with font color, etc.)

Multilingual String

Multilingual String properties may have a value for each Language installed in Innovator, allowing users to input data in multiple languages. The displayed value depends on the client regional laptop settings. Find more information in the *Configuring Internationalization* guide.

Note

Global Version and Unsigned BigInt are special data types used in synchronization processes. You can find more information about them in the *Data Synchronization Services Programmer's Guide*.



Property Settings

ItemType Property Settings are configured on the Properties tab.

Length

Length sets the maximum number of characters for string values.

Precision and Scale

Used for Decimal data type. **Precision** sets the maximum number of digits, and **Scale** assigns the number of digits after the decimal point.

Required

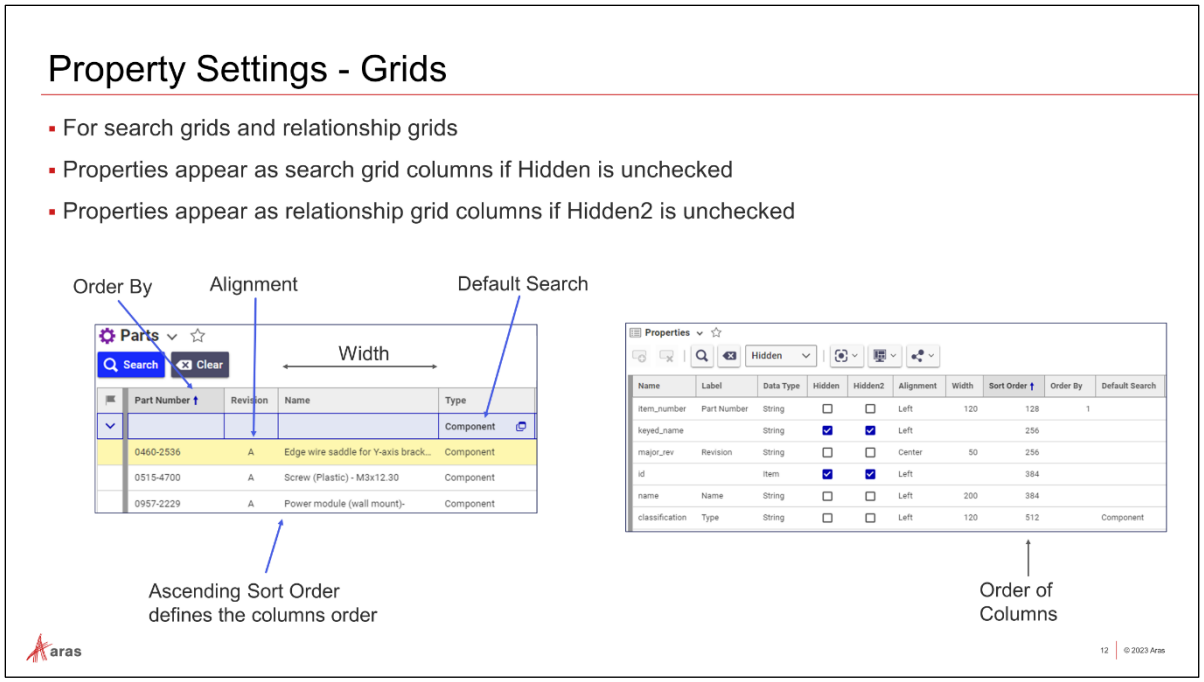
When the Required setting is selected, a property value will be required to save the item. This is enforced on save, meaning you cannot save an item that does not have a value for a required property. Do not set Required for Sequence properties (the value will be auto-assigned by the server).

Unique

The Unique setting enforces uniqueness on a property value. Setting Unique on multiple properties means that the combination of those properties must be unique. For example, you can set the combination of Part Number and CAGE Code (a manufacturer identifier) to be unique so that you can maintain unique parts from different manufacturers that just happen to have the same part number.

Indexed

The Indexed setting sets a database index on the property.



Property Settings - Grids

Hidden

The Hidden setting determines the property visibility on the Search Grid. If set to true, the property is not displayed in the main grid for this item.

Hidden2

The Hidden2 setting determines the property visibility on the Relationship Grid. If set to true, the property is not displayed on the relationship grid.

Alignment

Alignment determines the left/right/center alignment of the property in grids.

Width

Sets the width of the property column on the Search Grid (in pixels).

Sort Order

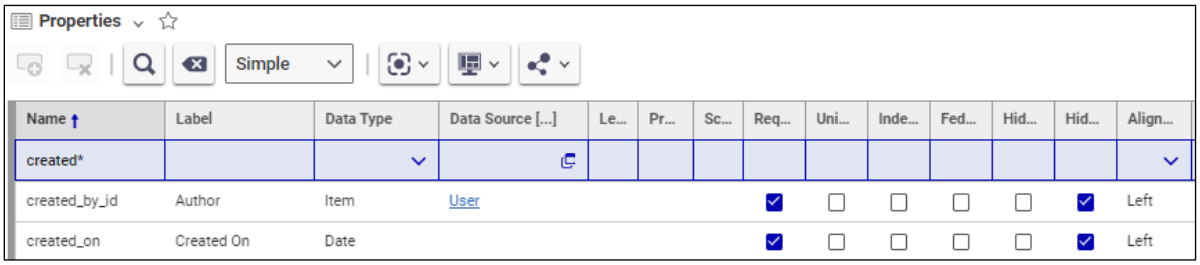
Sort Order determines the column order of the property on grids. Use integer values that control the relative position of columns in a search grid display with lower values displaying from left to right.

Order By

Order By controls the ascending sort order of the items displayed in the main item grid, after either a manual or an automatic search. Whichever property has the lowest value set with Order by, is the first one to control the order. For example, if you want to order the employees by last name and then by first name, last name will have a value of 1 in the Order By property, and first name would have a value of 2.

Try it … Modify some properties settings

1. Open the **Design Request** ItemType for editing.
2. Go to the Properties tab.
3. Locate the created_by_id and created_on system properties on the properties grid.
4. Uncheck the “Hidden” checkbox for both properties.
5. Add respectively “Author” and “Created On” in the Label column as well:



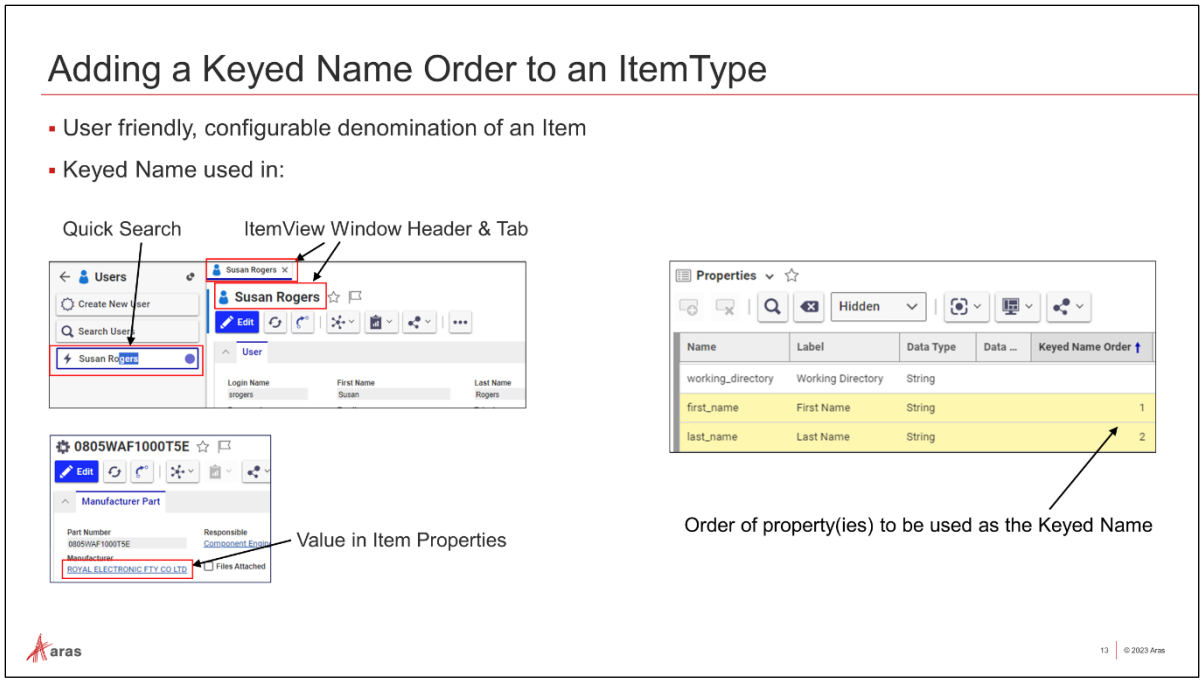
Name	Label	Data Type	Data Source [...]	Le...	Pr...	Sc...	Req...	Uni...	Inde...	Fed...	Hid...	Hid...	Align...
created*													
created_by_id	Author	Item	User				☑	☐	☐	☐	☐	☑	Left
created_on	Created On	Date					☑	☐	☐	☐	☐	☑	Left

6. Save the Design Request ItemType.

Try it … Make a Property Required, and Modify its Label

Before you can make a Property required, you must first delete any existing item instances having that Property with a null value.

1. If you have any Design Request items without an assigned value in the owned_by_id property, delete them now.
2. Open the **Design Request** ItemType for editing.
3. Select the **Properties Tab** and set the **owned_by_id** to *Required*.
4. Set the Label of **owned_by_id** to *Request Owner*.
5. Click Save.



Adding a Keyed Name Order to an ItemType

Keyed Name Order can be set on one or more properties to make it easier for Aras Innovator users to find objects stored in the database. By default, the internal Id of the Item is used for labels and searches. By setting a property, such as the name or an item number, as the keyed name order, users see descriptive information as labels and fields for easier searching.

To use multiple properties in the Keyed Name, enter integer values in each of the Keyed Name Order columns for each property desired to contribute to the keyed name. The integer value indicates the order that the property displays in the Keyed Name, for example on a Tab heading.

Keyed Name Order

Keyed Name Order controls the keyed name of an item. A keyed name is an alternate way to identify an item, for example on the Quick Search, on the title bar when opening the item, and the searches of Item Properties. A keyed name is comprised of one or more item properties separated by a space. If no keyed name for an ItemType is specified, the keyed name is the same as the Item id property (the GUID). The keyed name may use as many property values as necessary to identify the item in a meaningful way. You assign the order of the properties in the keyed name field using numbers (low to high) to control the sort in returned records. For example, a User item has two properties defined: first_name and last_name. To create a keyed name with the value of the last name [space] first_name, order the properties using the lowest number for property to display first, followed by higher numbers for additional properties.. A keyed name is set when the item is saved to the database.

Other Property Settings

- **Federated:** data is not stored in the Aras Innovator database, instead accessed from an external system
- **Default Value:** value auto-populated on the form for a new item
- **Pattern:** for Strings, expression syntax used for data validation. E.g. \d{3}-\d{3}-\d{4} for US phone numbers.

Also used for Filter Lists.

- **Class Path:** indicates if a property is specific to a certain classification of the Item.
- **Item Behavior:** for Item properties - defines which generation of a related Item is visible from the source Item.
- **Foreign:** displays the property value of a related Item in read-only

aras

14 | © 2023 Aras

Other Property Settings

Federated

Data marked as federated is not stored in the Aras Innovator Database. Instead, it is accessed from an external system. Reading and managing values of federated properties needs programming. Federated Properties do not have a column in the Aras Innovator database, but the definition of such properties is stored in a row in the Property table.

Please note that not every data type is supported for federated properties.

Default Value

When new items are created, the form will be populated with any properties including a default value, which must match the data type set. Users can change the value before saving. If left blank, it is stored as null.

Default Search

A Default Search is used to predefine the search criteria for an Item. If a user has not searched on the Item previously (no user preference), you can specify default search criteria for one or more properties. A user can override these values.

Pattern

A Pattern applies only to properties of type Date, String, or Filtered List.

If a property is of type Date, one of the following lists can be used as a Pattern. The order of display for month, day and year will follow internationalization and localization rules dependent on client settings.

- Short Date mm/dd/yyyy
- Short Date and Time mm/dd/yyyy hh:mm:ss AM/PM
- Long Date DayOfWeek, Month dd, yyyy
- Long Date and Time DayOfWeek, Month dd, yyyy hh:mm:ss AM/PM

If the property is of type String, the Pattern follows basic regular expression syntax. For example, to require that a user enters a telephone number in the format ###-###-#### you could use the regular expression: [0-9]{3}- [0-9]{3}- [0-9]{4}$

If the property is of type Filter List, then Pattern is the pointer to the property which defines the filter.

Class Path

The Class path property setting indicates that the property is only available to this subclass (and any additional subclasses of this subclass). A class structure must be added to the ItemType to enable this setting.

Foreign Property

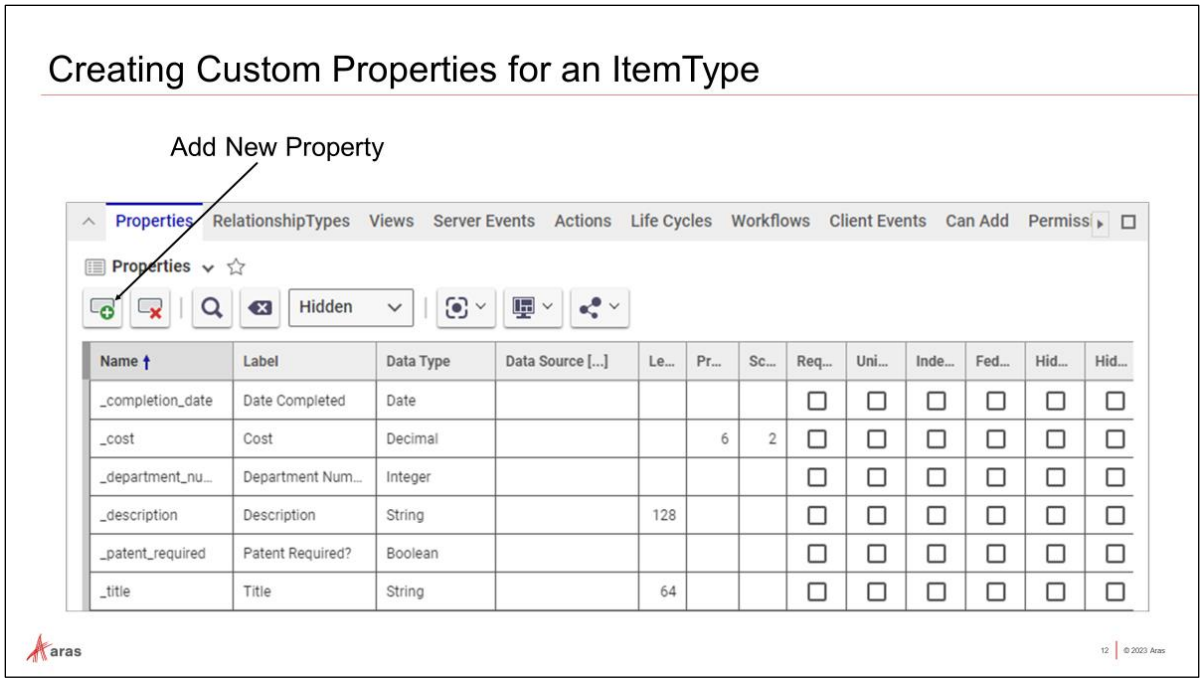
Foreign properties work with Item data type properties to allow you to refer to and display properties of another related Item. The Foreign Property stores the Property name of the referenced Item.

Track History

The Track History property setting indicates that the changed to and changed from values of the property will be stored in the Item's audit history depending on the ItemType History Template selected.

Item Behavior

Item Behavior refers to properties where the data type is Item, and the target item specified by the property is versionable. Item Behavior can be Fixed or Float.



Creating Custom Properties for an ItemType

When you save the Design Request ItemType, each new property automatically creates a column in the SQL database table. It is a good idea to consider using a prefix character(s) for a property name to distinguish it from the standard Aras system properties. In this example, we used the underscore character.

Try it … Create Custom Properties for the Design Request ItemType

1. Come back to the **Design Request** ItemType.
2. Select the Properties Tab and click the New Property button to add a new property.

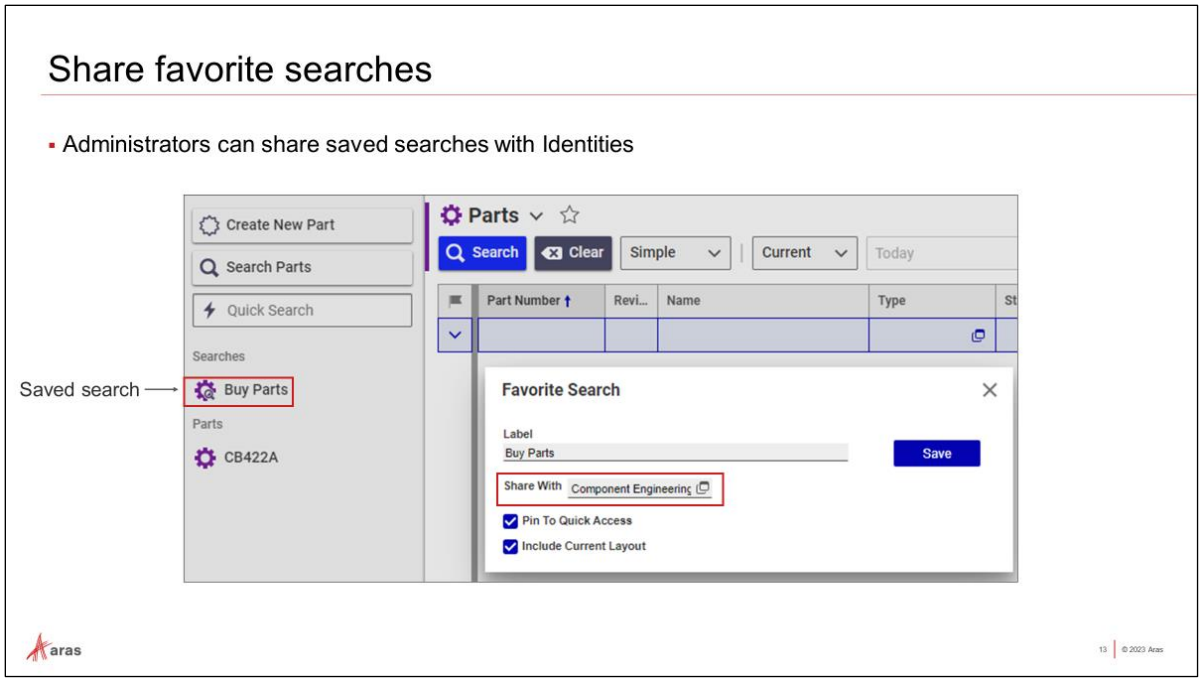
   Use the table below to provide the Name, Label, Data Type, and additional settings for each new property for the **Design Request** ItemType.

Property Name	Label	Data Type	Length	Precision	Scale	Width	Sort Order	Class Path	Default Search
_completion_date	Date Completed	Date							
_cost	Cost	Decimal		6	2				
_department_number	Department Number	Integer						Internal	
_description	Description	String	128			250	40		
_patent_required	Patent Required?	Boolean							0
_title	Title	String	64			150	20		

3. Locate the Classification Property. Enter *Customer* as the Default Value.
4. Click Save.

Try it … Explore the impact of search settings on the search grid

5. Navigate to Product Management > Development > Design Requests.
6. Open a search grid for Design Requests.
7. Notice that the columns refer to properties with *Hidden* unchecked on the ItemType. The column order follows the *Sort Order*. The Title and Description have a different column width, according to Width setting. Patent required has a pre-configured search criterion=0.

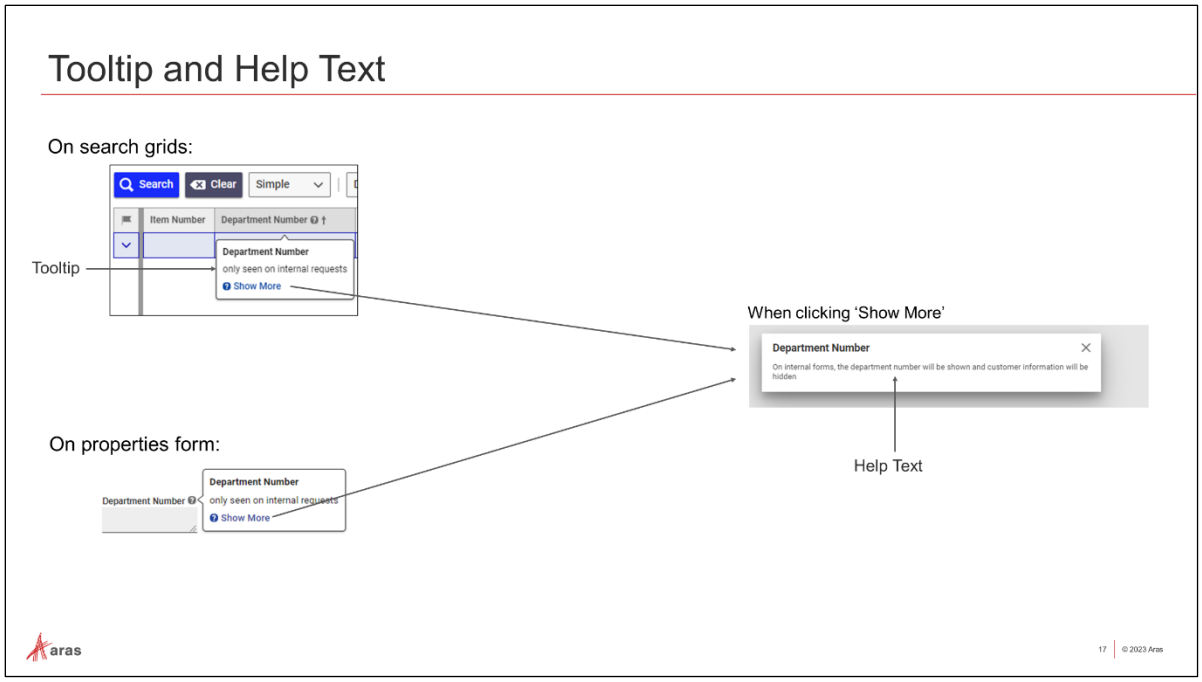


Share favorite searches

The administrators can share favorite searches with other identities, especially group identities, to help end users have a quick and relevant access to data. The saved searches will be invisible to users not belonging to the selected group identity.

Try it … Open existing saved searches

1. Open the Parts Item Panel.
2. Run several saved searches, e.g., *Buy Parts* and *In Work Parts*. Notice that both simple searches and advanced searches are saved.



Tooltips for Properties in Grids and Forms

The standard attributes "Tooltip" and "Help Text" of ItemType Properties may be configured to display contextual help information for properties displayed as grid columns and form fields.

Setting a value for "Tooltip" will display a "?" icon beside the column/field label. Hovering the cursor will display the tooltip value.

Additionally, all grid column headers will display a simple browser tooltip on hover containing the column label so truncated labels can be viewed without needing to expand the column width.

Setting a value for "Help Text" will add a "Show More" link inside the tooltip which, when clicked, will display a dialog window for longer text descriptions. Note that you can have a Tooltip value without a Help Text value.

Try it … Open the Design Request Main Search Grid to View Properties

3. On the **Design Request** ItemType, add the following values to the _department_number property:

Property Name	Tooltip	Help Text
_department_number	Only for Internal Requests	When using a form for internal requests, no customer information will be shown

4. Click Done on the Design Request ItemType.
5. Navigate to Product Management > Development in the TOC.
6. Open a new search grid for Design Requests.
7. Inspect the Department Number column header to see the Tooltip and Help Text.

Summary

In this unit, you learned how Properties are configured and how to build Custom Properties for an ItemType.

- You should now be able to:
- Describe the difference between System Properties, Custom Properties and Extended Properties
- Create a Custom Property for an ItemType
- Identify Property Settings
- Use the Property Data Types for different situations.

Review Questions

1. Where are Custom Property definitions stored in the database?

2. Where are Custom Property values stored in the database?

3. What is the purpose of the Hidden and Hidden2 settings on a property?

4. What is the purpose of the Label column?

5. What is the purpose of Keyed Name Order?

6. Describe the differences between System Properties and Custom Properties.

Lab Exercise

Goal

Be able to create custom properties for an ItemType and understand where the property data gets stored in the database.

Scenario

In this exercise, you make a property required and create custom properties for the Change Request ItemType.

Steps

Before you can make a Property required, you must first delete any existing items because their values for the Property are null.

1. Open the **Change Request** ItemType for editing.
2. Set the **owned_by_id** label to *Owner.*
3. Select the Properties Tab and set the **owned_by_id** to **Required**.
4. Click Save.
5. Add the following properties to the Change Request ItemType:

Property Name	Label	Data Type	Length	Precision	Scale	Width	Sort Order	Class Path
_title	Title	String	64			150	40	
_reason	Reason	String	256			250	60	
_target_date	Target Date	Date						
_cost_estimate	Cost Estimate	Decimal		6	2			
_defect_details	Defect Details	String	128					Defect

6. For the **state** property, uncheck the Hidden checkbox and set the Sort Order to 20.
7. Click the Done button to save the Change Request ItemType and release your claim.

This page intentionally left blank

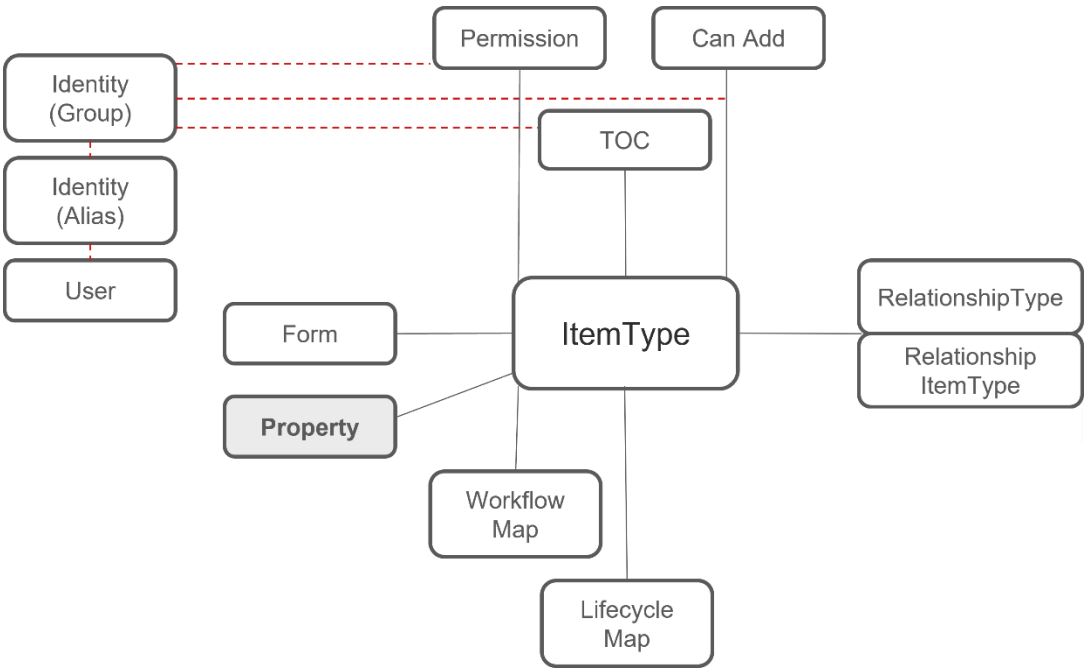
Unit 6 Creating Object Properties

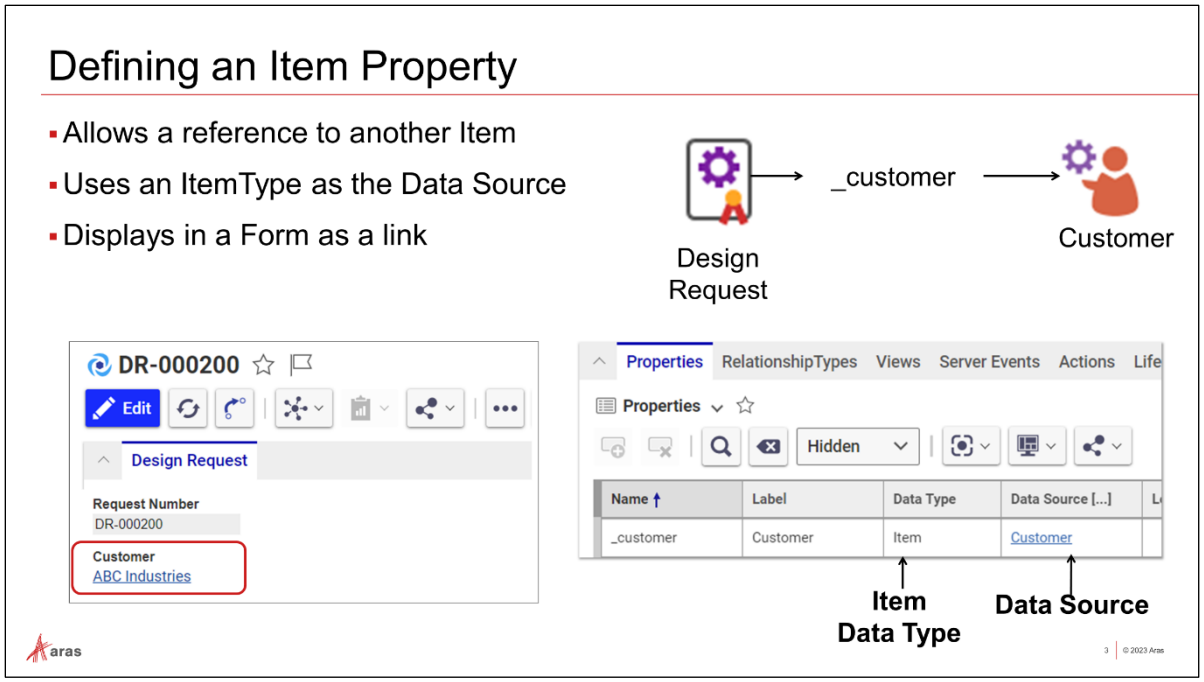
Overview

In this unit, you create properties of several different data types. Using these different data types can provide a more robust user interface. Users will be able to populate values using searches, by selecting from a dropdown list and even browse for an image. Some of these data types even auto-populate data, saving users from having to enter data manually.

Objectives

- Define an Item Property
- Define a Foreign Property
- Define a List Property
- Define a Filtered List Property
- Build a Sequence
- Use an Image Property





Defining an Item Property

Item Properties allow you to reference another Item as the property value. Instead of storing a value of a particular data type (E.g., Date, String, List, etc.) in the database, an Item Property stores the ID of a referenced Item. It is displayed as a field with a lookup icon to the right when editing, or as a hyperlink when in view mode. An Item property must have a source data type specified in the Data Source column in the Properties grid.

In the example above, the Design Request Item has a property named **_customer** whose value is a pointer to a Customer Item. This specific Design Request has a one-to-one relationship with the Customer Item.

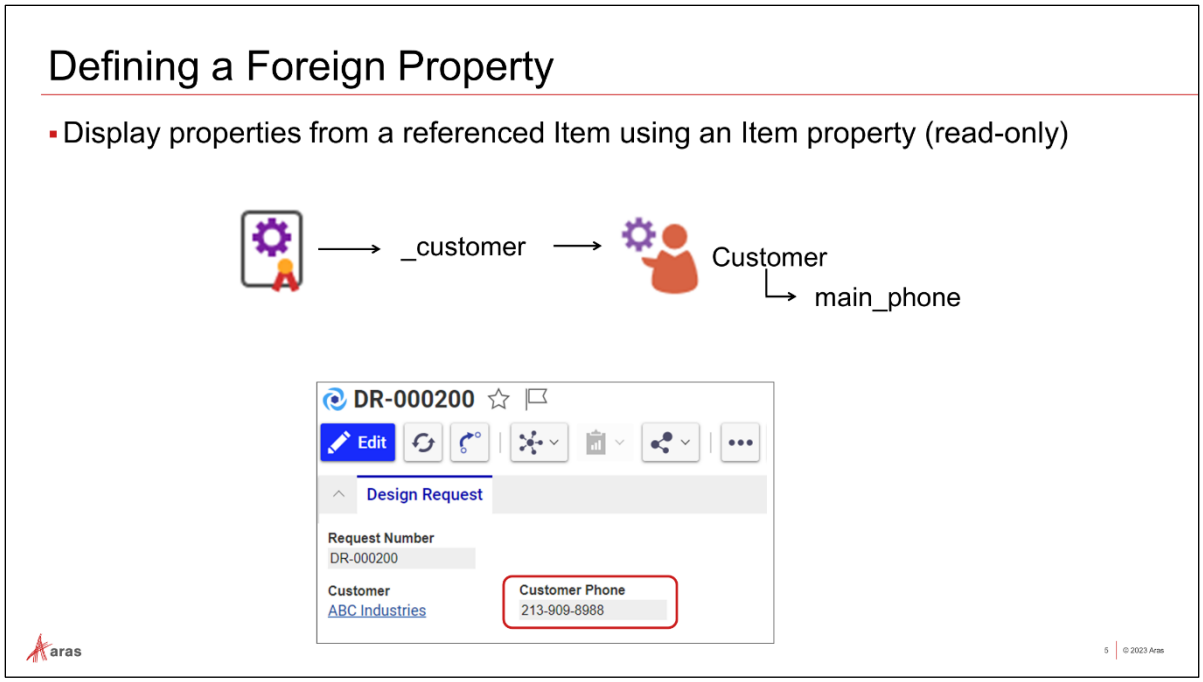
The Keyed Named Order of a referenced Item is important. The search grid of the Item Property will display the Keyed Name of the referenced Item; but if a Keyed Name is not defined the internal Item ID (GUID, a 32-digit hexadecimal number) will be displayed instead.

Try it … Create an Item Property

1. Navigate to Administration > ItemTypes and open the **Design Request** ItemType for Editing.
2. On the Properties Tab, create a new property by clicking the New Property button.
3. Enter the following:

Name	Label	Data Type	Data Source […]
_customer	Customer	Item	Customer

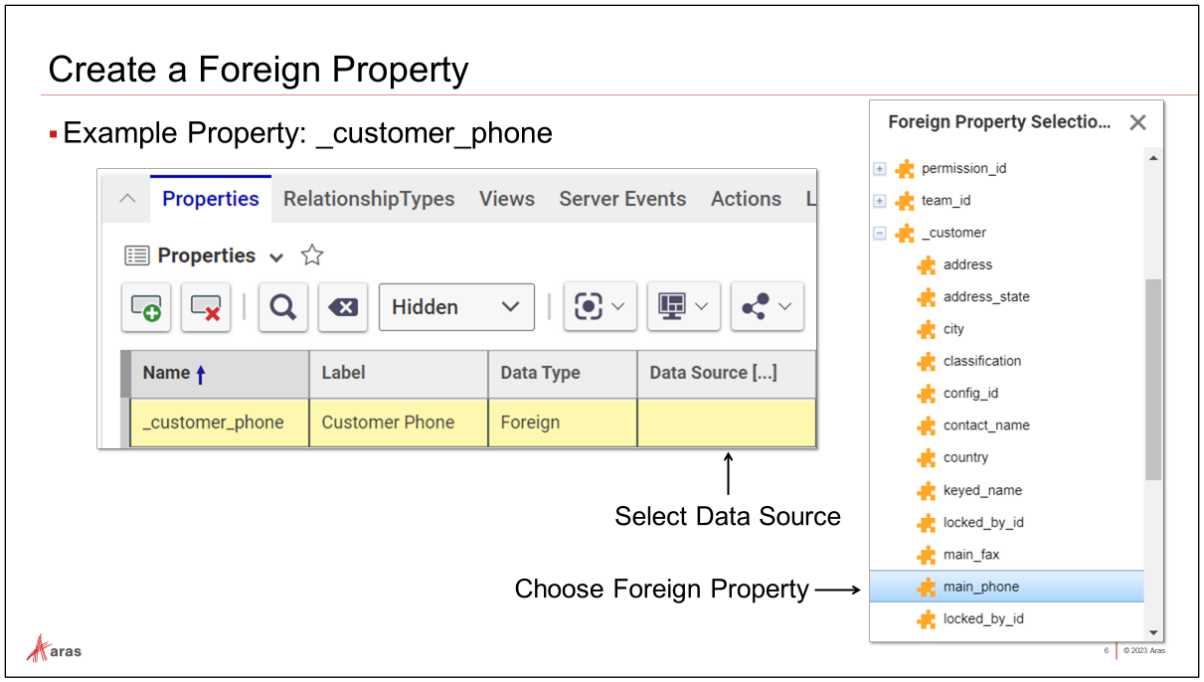
   For the Data Source column, either use the Type Ahead feature or press the F2 (lookup) key to search and select.
4. Click the Save button.



Defining a Foreign Property

Foreign properties can be created after an Item Property is defined, to include additional property information from the referenced item (in the above example, one property from the referenced Customer Item). The foreign properties can be made visible on the form but display as read-only.

Multiple foreign properties can be created to display more properties from the Item selected in the Item Property.



Create a Foreign Property

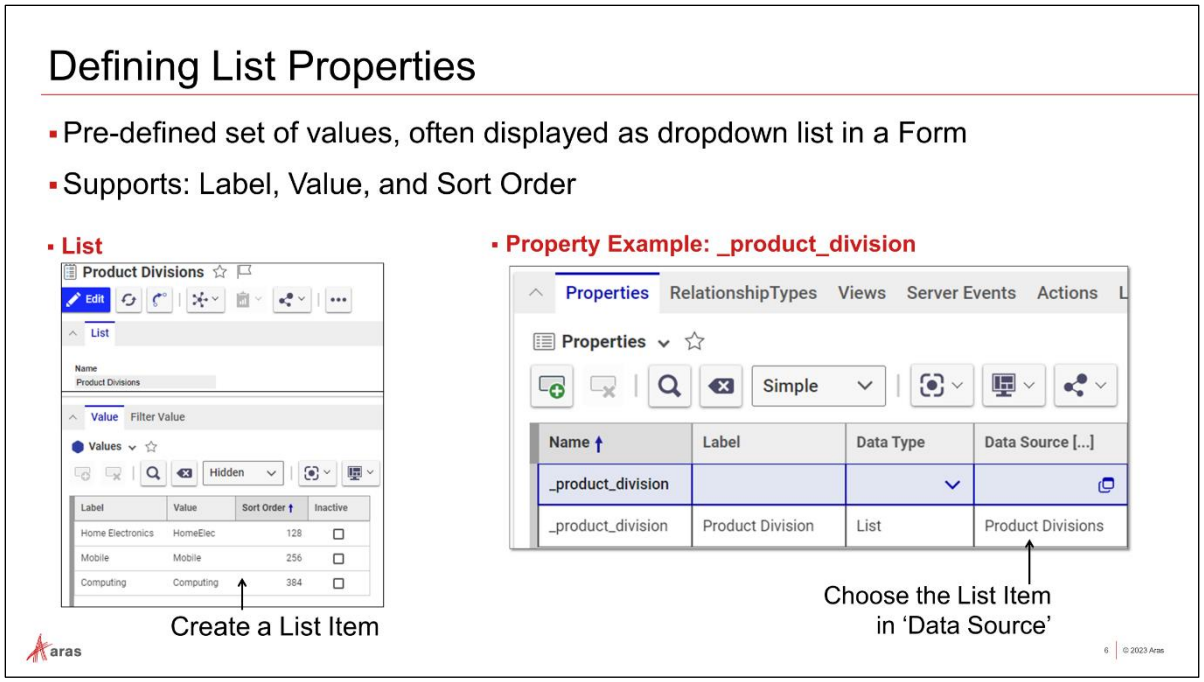
Foreign properties display in the Foreign Property column in the relationship grid after saving the ItemType.

Try it … Create a Foreign Property

1. On the Properties Tab of the Design Request ItemType, click the New Property button.
2. Enter the following:

Name	Label	Data Type
_customer_phone	*Customer Phone*	*Foreign*

3. In the Data Source column, click to expand the properties available under ***_customer***.
4. Double-click on **main_phone** to select this as the foreign property.
5. Click Save.
6. Scroll to the right to see the foreign property name in the *Foreign Property* column.



Defining List Properties

Lists can be used anytime that there is a predetermined group of things to select from. This allows users to select only valid items that have been predefined. The default display of a list is a dropdown menu, but it can be presented as other options when placed on a form.

Administrators have full control of defining what text will display (Label column) when the List is rendered. They also decide what will be stored within the SQL table once a selection is made (Value column). All stored values must be unique so attention to detail is critical when constructing Lists. The order of how Items will appear is controlled by setting ascending numbers in the Sort Order column. Once a selection is no longer valid for future use, the row should be set to Inactive. Inactive labels will not display for future selections, but the inactive value is kept for traceability, especially if existing items use that value.

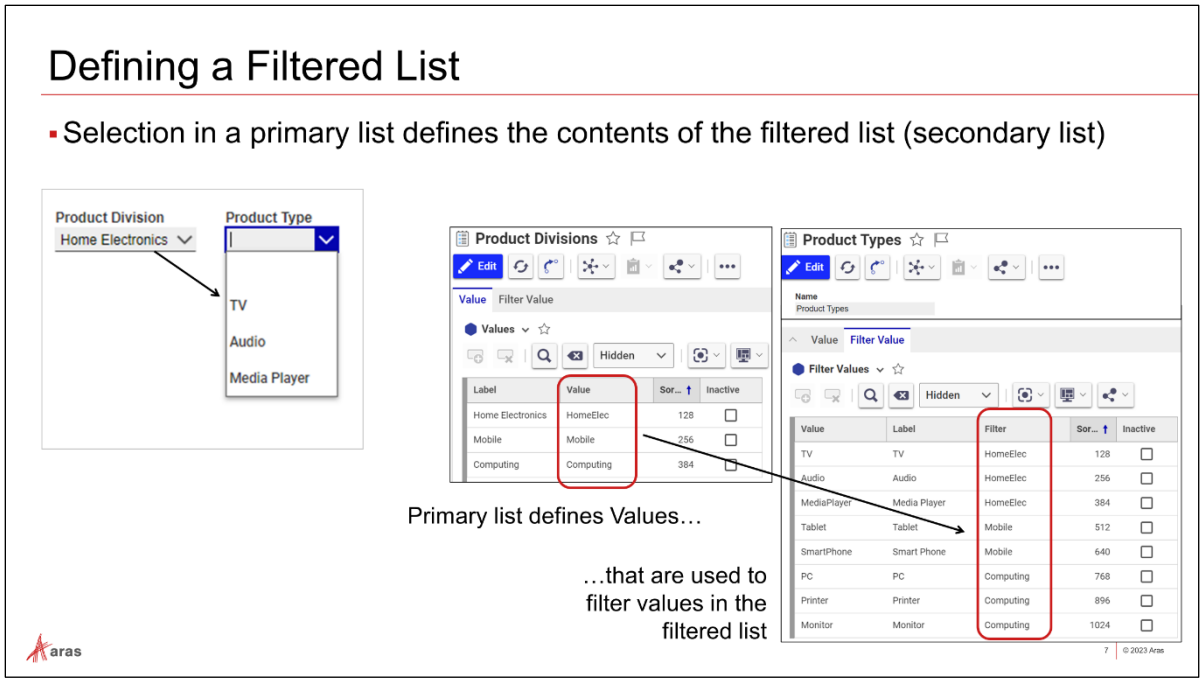
Lists will also display as dropdown lists when used to filter on any search grid.

Try it … Create a List and use it for a new Property

1. Navigate to Administration > Lists and create a new List Item. In Name property, enter *Product Divisions*.
2. On the Value relationship tab, click the New Value button and enter the values shown in the table below.

Label	Value
Home Electronics	HomeElec
Mobile	Mobile
Computing	Computing

3. Click the Done button. The Sort Order column gets updated automatically by increments of 128.



Defining a Filtered List

A filtered list allows you to narrow the selection of choices in that filtered list (secondary list) based upon selecting an option from the first or primary list. A filtered list is a separate list where the Filter Value tab is configured instead of the Value tab on a list. An extra column (Filter) is added to specify the Value from the primary list that this secondary list should be filtered against.

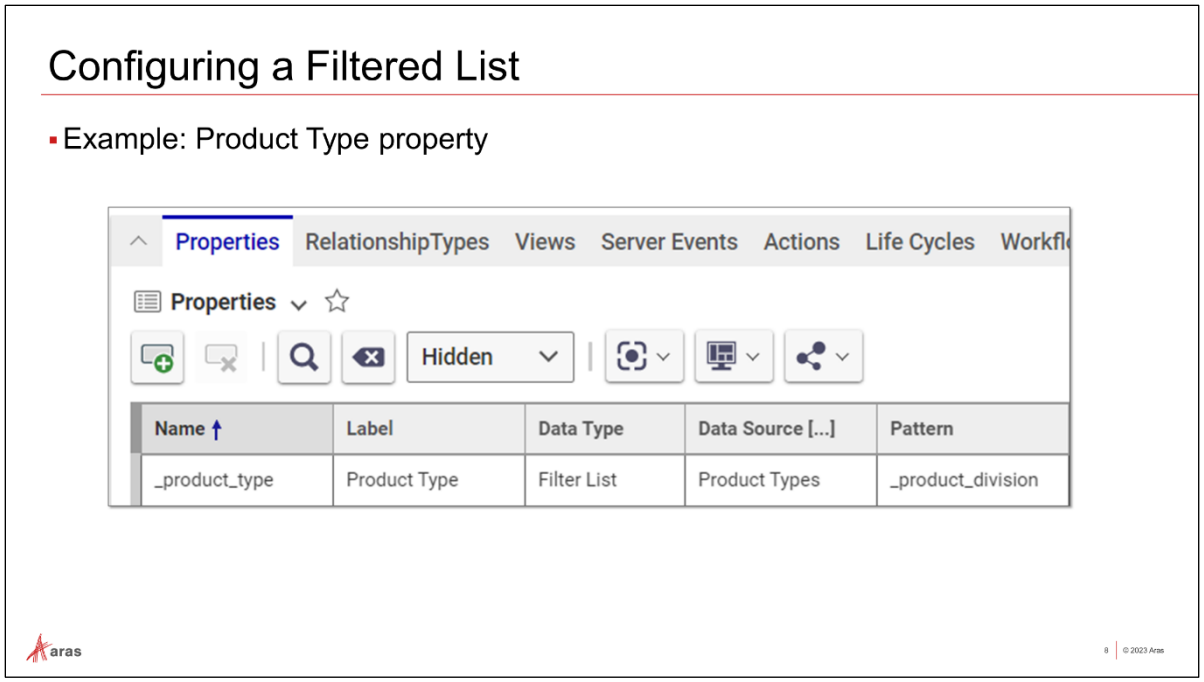
Note: for the primary list, use the List No Blank setting on classic Forms' Field Type tab. This avoids mistakes or confusion during data entry on the secondary list.

Try it … Create a Filtered List

In this exercise, you use the list you created previously as a primary list to filter this filtered list.

1. Create another List named *Product Types*.
2. Go to the **Filter Value** relationship tab.
3. Click the New Filter Value button, enter the values shown in the table below, then click Done.
   Note: the Sort Order will be filled automatically upon clicking Done.

Value	Label	Filter
TV	TV	HomeElec
Audio	Audio	HomeElec
MediaPlayer	Media Player	HomeElec
Tablet	Tablet	Mobile
SmartPhone	Smart Phone	Mobile
PC	PC	Computing
Printer	Printer	Computing
Monitor	Monitor	Computing



Configuring a Filtered List

Try it … Create List and Filter List Properties

In this exercise, you first configure the List Data Type, then you configure the Filter List data Type.

1. Return to the **Design Request** ItemType and click the New Property button.
2. Enter the values for the new property, as follows:

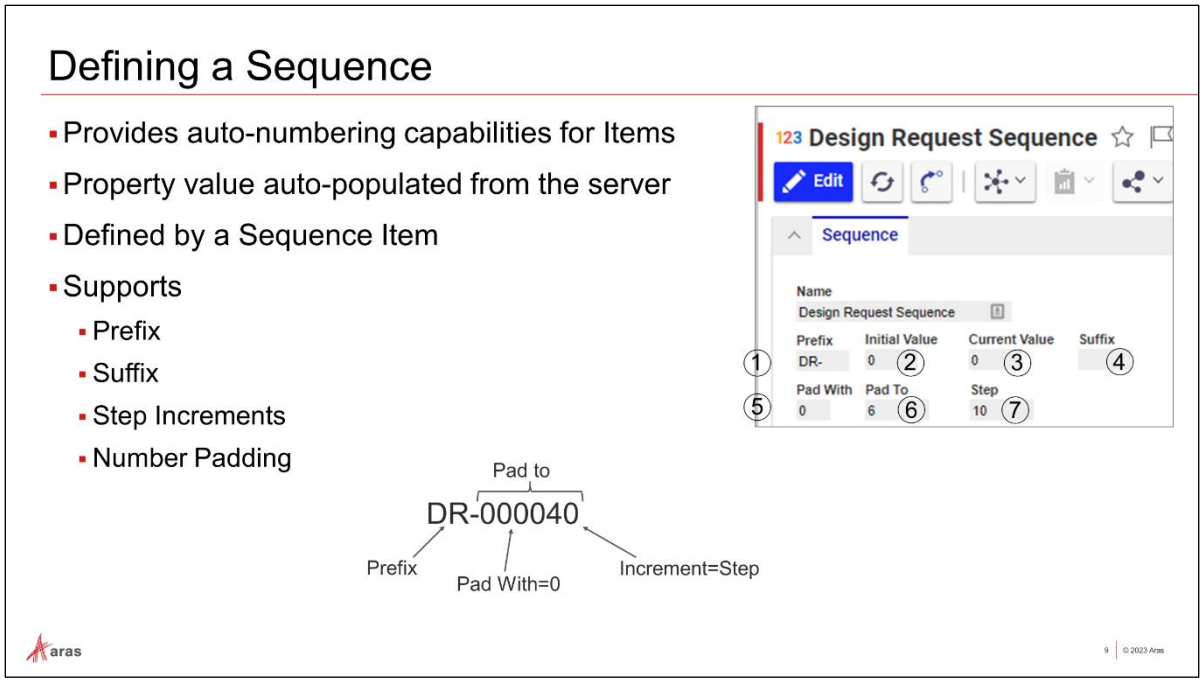
Name	Label	Data Type	Data Source
_product_division	Product Division	List	Product Divisions

3. Click the New Property button again.
4. Enter the new property values, as follows:

Name	Label	Data Type	Data Source	Pattern
_product_type	Product Type	Filter List	Product Types	_product_division

Note: you will need to scroll to the right to find the Pattern column.

5. Click Save.



Defining a Sequence

Sequences provide auto-numbering capabilities for Items. Properties of data type Sequence are populated on the server when the user saves the item after the check for required properties.

It is not necessary to mark a Sequence Property as required, since the server populates a Sequence Property only after having checked the existence of required Properties.

The following fields are used to create a Sequence:

①	**Prefix**	Up to 16 characters that will prefix each sequence number.
②	**Initial Value**	This value is used to set the Current Value or import to another Aras database if the Current Value field is NULL.
③	**Current Value**	Current sequence number value.
④	**Suffix**	Up to 16 characters that will follow each sequence number.
⑤	**Pad With**	Padding character to use (if blank no padding characters display).
⑥	**Pad To**	Total number of characters in the sequence number (excluding prefix and suffix).
⑦	**Step**	Increment of sequence number (typically 10).
⑧	**Set Data Source**	Add a new property to an ItemType and specify Sequence as the Data Type.

Try it … Create a Sequence

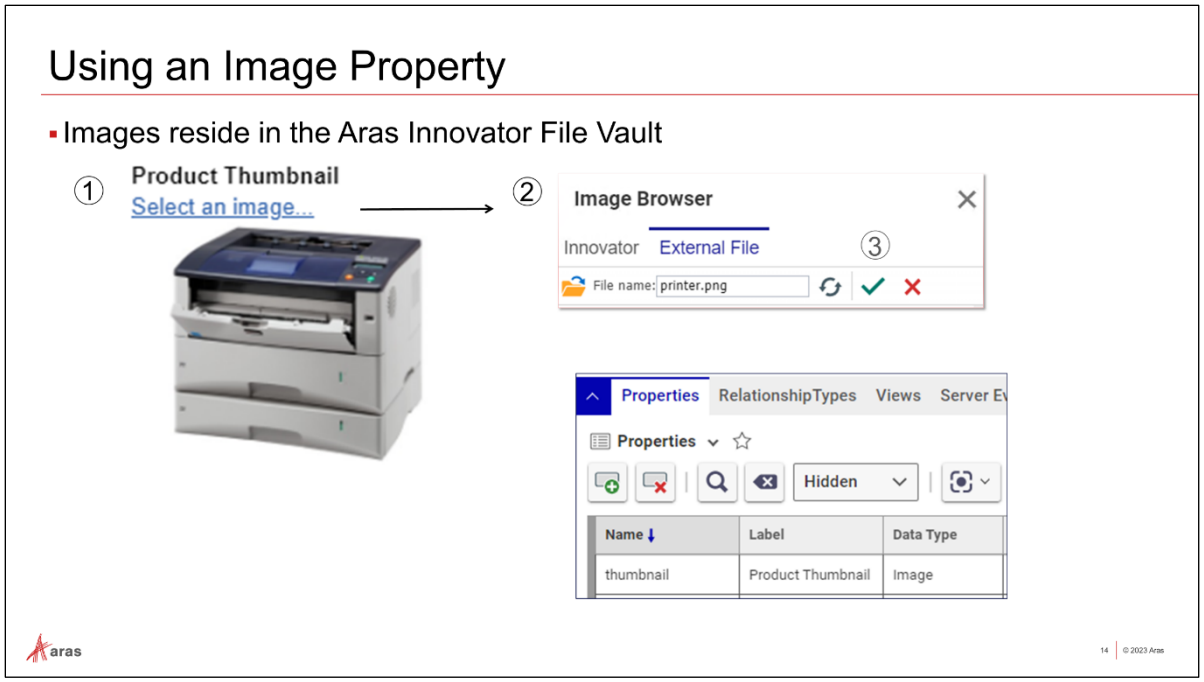
1. Navigate to Administration > Sequences and create a new Sequence.
2. Name the sequence *Design Request Sequence*.
3. Provide the following sequence values:

Prefix	DR-
Initial Value	0
Current Value	0
Suffix	
Pad With	0
Pad To	6
Step	10

4. Click Done.
5. Return to the **Design Request** ItemType tab and ensure you are in Edit mode.
6. On the Properties Tab, click the New Property button.
7. Add a new property as follows:

Name	Label	Data Type	Data Source	Width	Sort Order	Keyed Name Order	Order By
_item_number	*Request Number*	*Sequence*	*Design Request Sequence*	*150*	*10*	*10*	*10*

8. Click Save.



Using an Image Property

The Image Property allows a user to select an image file for display on an Item.

Users can select an external image file from any available location. Once the selected image file is saved in Aras Innovator, it is automatically checked into the user's Default Vault location for later use. This can be useful for storing thumbnails, diagrams, etc. associated with an Item.

A new Property with the Data Type Image must be created in such case. When this property is to a Form for display, the size of the image can be adjusted by entering a width and height on the Field Physical tab of the Form. If those settings are blank, the image displays using its original dimensions.

If the Image property is named ‘thumbnail,’ then the image displays when selecting Display > Properties Preview in the Item search grid.

Try it … Defining an Image Property

1. On the Properties Tab, click the New Property button.
2. Add a new Property as follows:

Name	thumbnail
Label	Product Thumbnail
Data Type	Image

3. Click Done.

Summary

In this unit, you learned how to create properties that use another Item as their value and how to access the Properties of this other Item. You also learned about Lists and Sequences.

You should now be able to:

- Define Item Properties
- Define a Foreign Property
- Define List and Filtered List Properties
- Build a Sequence
- Use an Image Property

Review Questions

1. Foreign Property values can be edited by an end user in a Form. (True or False?)

2. What is required before I can add a Foreign Property to an ItemType?

3. How can an administrator reset a sequence number?

Lab Exercise

Goal

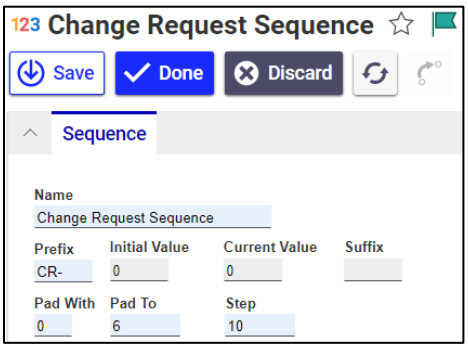
Add Properties to an ItemType that uses another Item as the Data Source. Add additional Data Type properties to an ItemType that allows automatic sequencing, selection from a list, and access to Foreign Properties.

Scenario

You define an automatic sequence number for Change Requests when they are created and add some additional properties that present a list to a user. You also define Properties linking a Change Request with a Document and display the name of the document to the user.

Steps

1. Create a new sequence named *Change Request Sequence* that includes the following settings:



2. Create a new List named *Regions*. The List should display the following:

Label	Value	Sort Order
Americas	Americas	128
Asia	Asia	256
Europe	Europe	384

3. Create a new Filtered List named *Design Centers*. The filtered list should be tied to the Regions list you built above. As this second list is a filtered list, please make sure to populate the second tab 'Filter Value.' The Filtered List should respond as follows:

Value	Label	Filter	Sort Order
NewYork	New York	Americas	128
SaoPaulo	Sao Paulo	Americas	256
Shanghai	Shanghai	Asia	384
Bangalore	Bangalore	Asia	512
London	London	Europe	640
Frankfurt	Frankfurt	Europe	768

4. Open the **Change Request** ItemType for editing and add the following properties:

Property Name	Label	Data Type	Data Source	Sort Order	Keyed Name Order	Pattern
_item_number	Request Number	Sequence	Change Request Sequence	10	10	
_region	Region	List	Regions			
_design_center	Design Center	Filter List	Design Centers			_region
_attachment	Document	Item	Document			

5. Add a Foreign Property named *_attachment_name* that will be used to display the name of a selected document based on the _attachment property above.
   (HINT: In the Data Source dialog, click the plus sign (+) next to _attachment and select **name** as the Foreign Property.)
6. Click Done.

This page intentionally left blank.

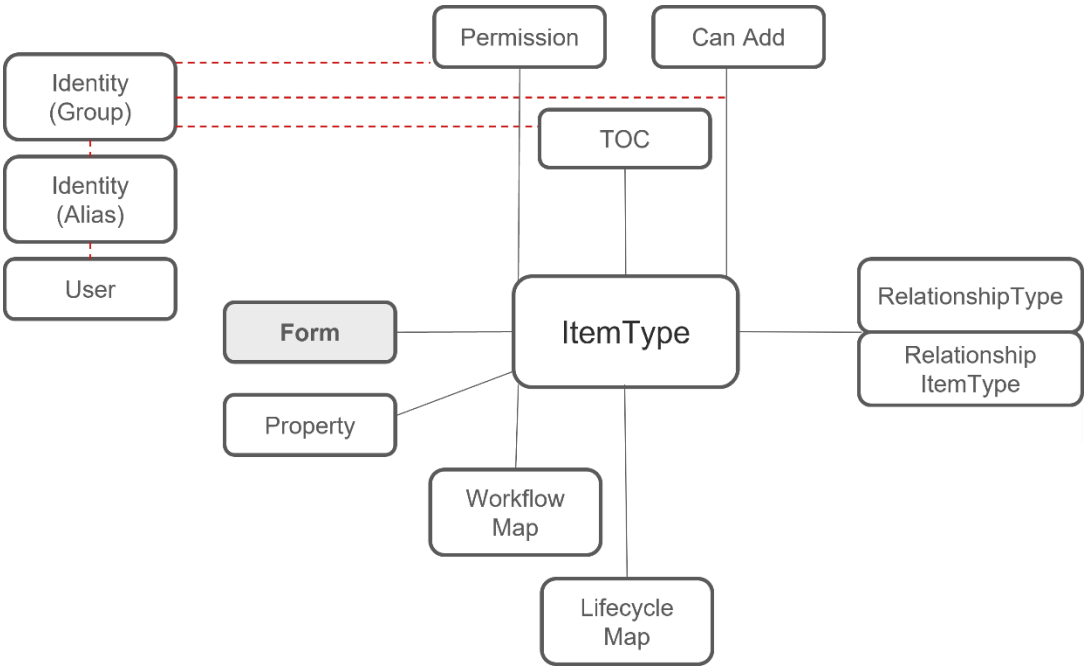
Unit 7 Creating the User Interface

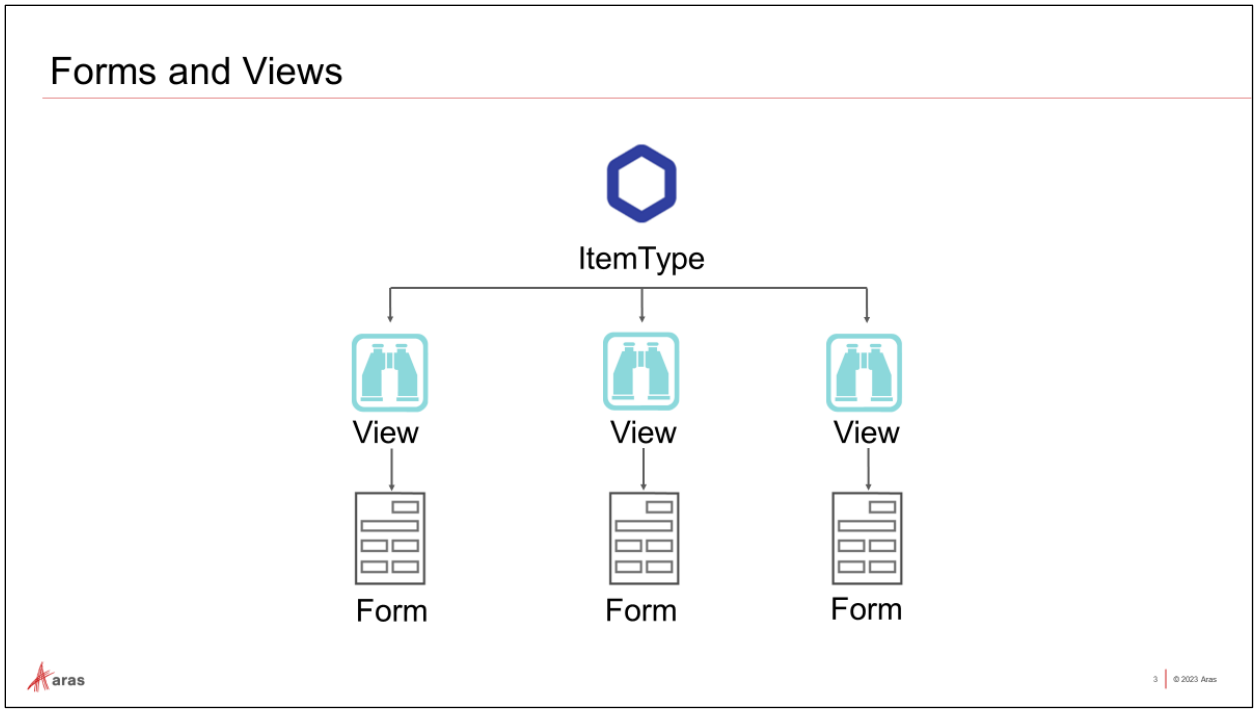
Overview

In this unit, you will create a user interface for a solution that allows a user to interact with web Forms.

Objectives

- Understand Forms and Views
- Create a User Interface using the Form Editor
- Review Field Settings
- Review Classic Form Settings
- Build an ItemType View
- Build a responsive Form



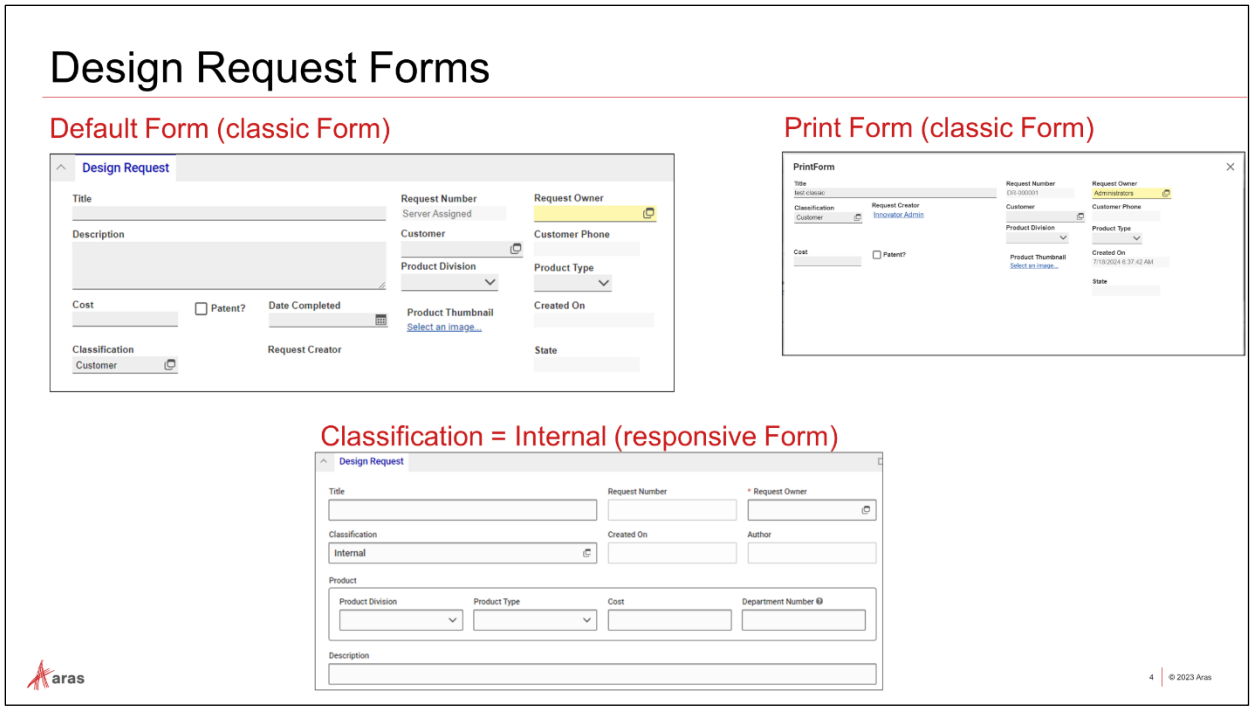


Forms and Views

A Form is a User Interface for an ItemType to allow viewing and editing of an Item instance of that type, provided that the User accessing the Form is authorized to do so.

A View is the relationship between an ItemType and a Form. Forms are designed using the Form Editor, and then associated to an ItemType on the Views Relationship Tab.

The same Item can be displayed to different users in different ways by creating multiple Views and Forms. One Form could be used for editing, while another Form could be used for printing.



Design Request Forms

In our example, we create three Forms for the Design Request solution:

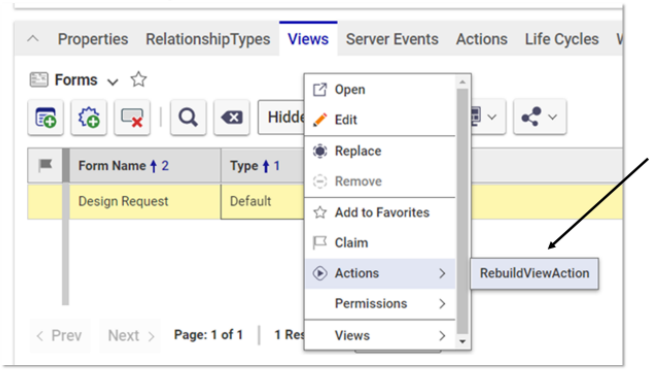
- One Form will display by default.
- A second Form will display only when printing an Item (share > Export to PDF).
- The third Form will display when a Design Request Classification value is set to Internal.

The different Forms show the custom properties defined on the ItemType, but only the default Form will show the Customer Name and Customer Phone Number properties. Only the Design Requests classified as Internal will show the Department Number property.

The default Form will be a classic Form with absolute positioning of the fields in X and Y, whereas the Internal Form will be a Responsive Form, i.e. the fields width and positioning will adapt to the user's screen size.

Regenerating a Form – Classic form

- A classic Form can be regenerated using an Action
- RebuildViewAction should only be used on **new** classic Forms – current layout will be replaced
- Do not use RebuildViewAction on responsive forms



Regenerating a Form – Classic Form

A Form with the same name as the ItemType is automatically created when the ItemType is created and saved in the database. Like any new Form, that Form is a classic Form by default. You can manually add Custom ItemType properties to Forms using a Form Editor tool, but we can also run an Action to speed up the administrator work. Indeed the RebuildViewAction resets the classic Form and adds automatically on that Form any non-hidden Properties defined for the ItemType.

Use the RebuildViewAction only on **new** classic Forms, as this Action removes any previous configuration on the Form. Make sure the Form is not claimed before attempting to regenerate it.

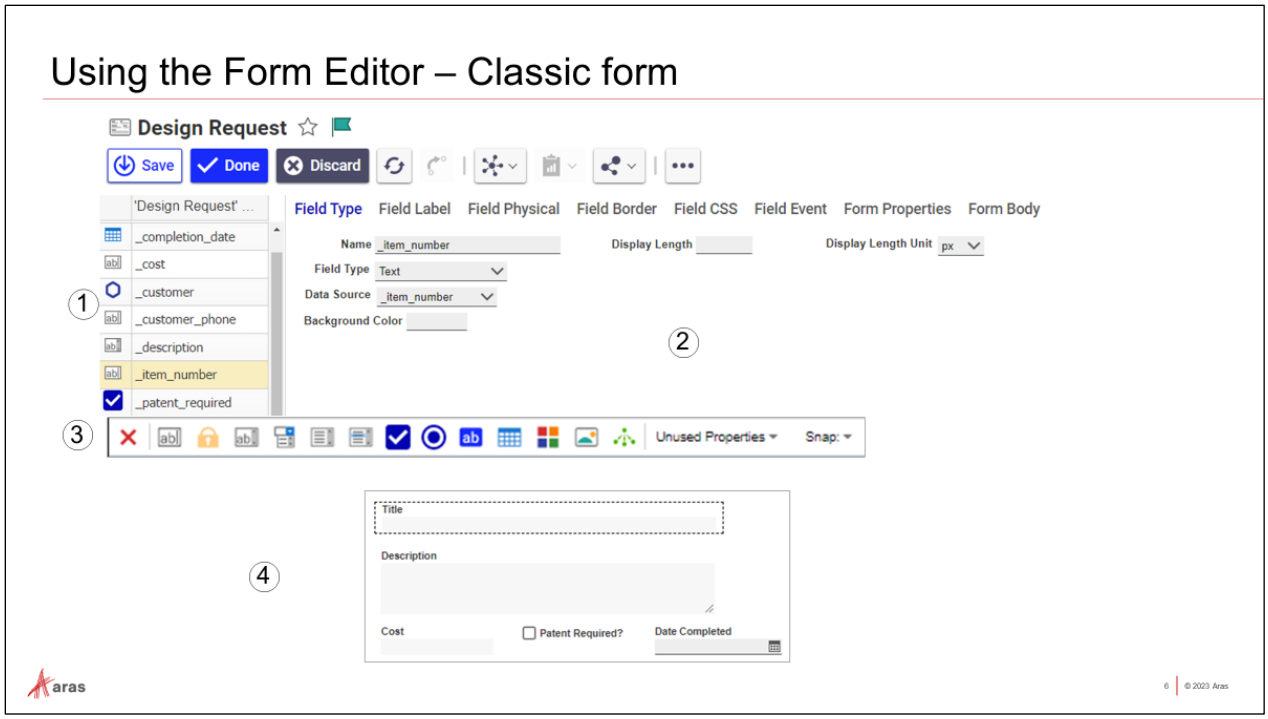
You can reposition Fields on a classic Form by dragging them or using X and Y coordinates. Field characteristics can be changed using the Field tabs.

RebuildViewAction cannot be used on a responsive Form, because responsive Forms consider all properties as hidden properties to be dragged-and-dropped manually.

Try it … Regenerate a Form

We want to regenerate the Form so the new properties we recently created are displayed.

1. Navigate to Administration > ItemTypes and open the **Design Request** ItemType for viewing.
2. On the Views relationship tab, right-click the Design Request (Default) Form and select Actions > RebuildViewAction from the context menu.
3. Right-click the Design Request (Default) Form again and select Open to open the Form Editor to see the Custom Properties have been added.



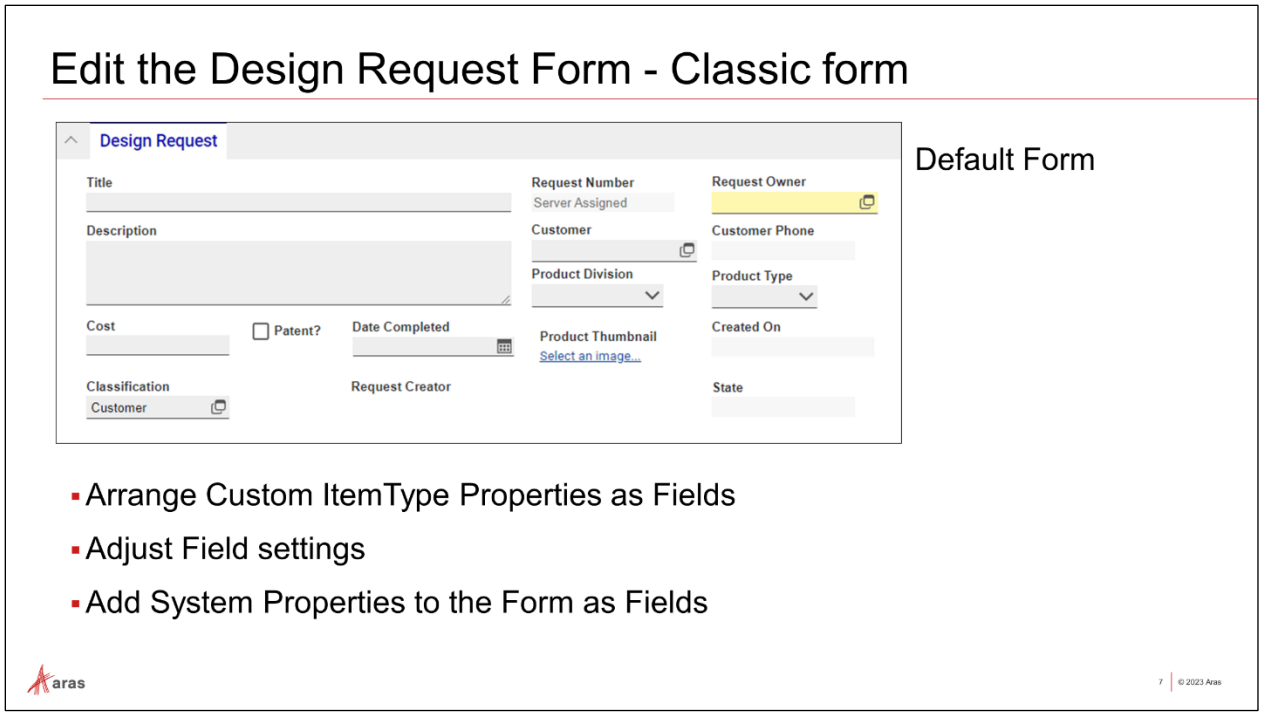
Using the Form Editor – Classic Form

Forms provide the main user interface for viewing and editing items.

- Forms are defined using the Form Editor, stored in the database, and they are rendered dynamically at run time.
- Forms may also incorporate custom HTML and JavaScript.

The Form Editor is divided into 4 areas:

①	**Properties**	Properties that have been added to the Form.
②	**Field and Form Definitions**	Specific information about the Form and each Field on the Form.
③	**Empty Fields Toolbar**	A series of Controls you can add to a Form – you can then associate the field to a property. This is where you can also add the xClass Tree Hierarchy for Extended Classification.
④	**Form Layout**	A drag and drop area to design the Form the user sees at runtime.



Edit the Design Request Form – Classic Form

Consider using the Required property setting for the Primary Lists to enhance the user experience.

Note

If during form edition, a field disappears, and instead a message "undefined" appears on the top-left corner of the form preview, then save the form. The field will appear again on the previous location.

Try it … Edit the Design Request Form

The Form Editor should already be open. If not, open the Design Request Form.

1. In the Form Editor, click Edit.
2. Select the **Title** Field and drag it to position it on the Form, as shown above, moving other properties as necessary.
3. With the **Title** field selected, click the **Field Type** tab and change the **Display Length** to *350*.
4. With the **Title** field selected, click the **Field Label** tab and notice how you can change the appearance of the Field label.
5. With the **Title** field selected, click the **Field Physical** tab. Ensure that the Tab Stop checkbox is checked and set the **Tab Index** to *10.*
6. Drag and drop to organize the Design Request fields as shown on the slide above. Note that some fields will need their labels to be edited, as shown in the following table:

Field	Label	Tab Stop	Tab Index
Description		checked	30
Cost		checked	70
Patent Required	Patent?	checked	80
Date Completed		checked	90

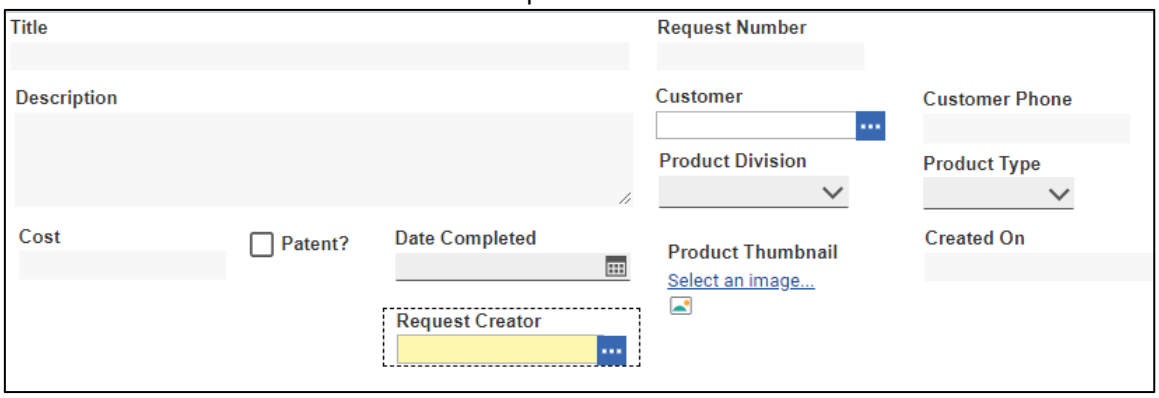
Try it … Configure Additional Properties on the Form

In the following exercise, the Product Division field (List) is set to **List No Blank** in the Form editor; otherwise, the filtered list results may be unpredictable.

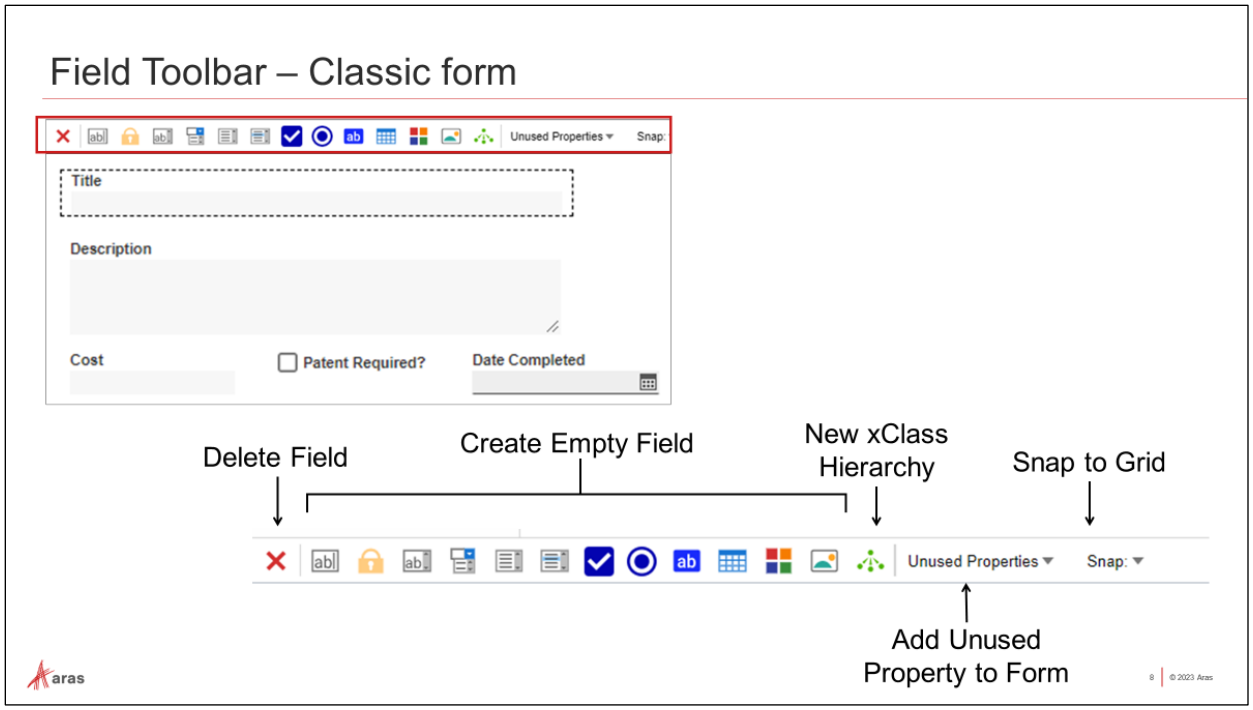
1. Continue arranging properties as shown above. Configure additional field settings as noted.

Property	Field Label	Tab Stop	Tab Index	Additional Field Settings
_item_number	Request Number	unchecked	None	
_customer	Customer	checked	40	
_customer_phone	Customer Phone	unchecked	None	
_product_division	Product Division	checked	50	On the Field Type tab, enable the List No Blank checkbox.
_product_type	Product Type	checked	60	
thumbnail	Product Thumbnail	checked	100	Set Image Size to Width = 120 and Height = 120.
created_on	Created On	unchecked	None	
created_by_id	Request Creator	unchecked		Change the field label to 'Request Creator'

Your Form should look similar to this at this point:



2. Click Save.



Field Toolbar – Classic Form

Use the Field Toolbar to add empty field controls to a Form. If you use the Unused Properties option, Fields are added automatically with the Property Data Type. You can add the following fields:

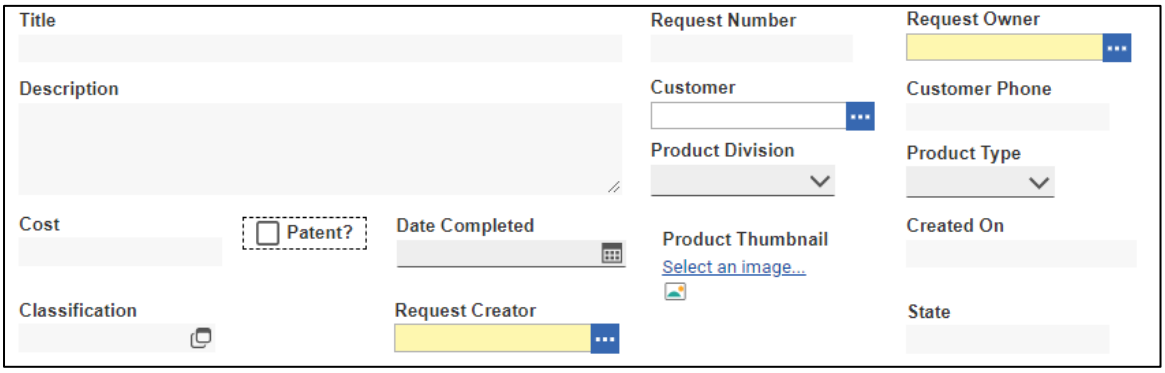
	Text field		
	Password field		New Button
	Text Area Field		Date field
	Dropdown Field		Color field
	Single Select List Box field		HTML field
	Multi Select List Box field		xClassification control
	Checkbox field		Unused Properties Dropdown
	Radio Button field		Snap to Grid

Try it … Add Unused Properties to the Form

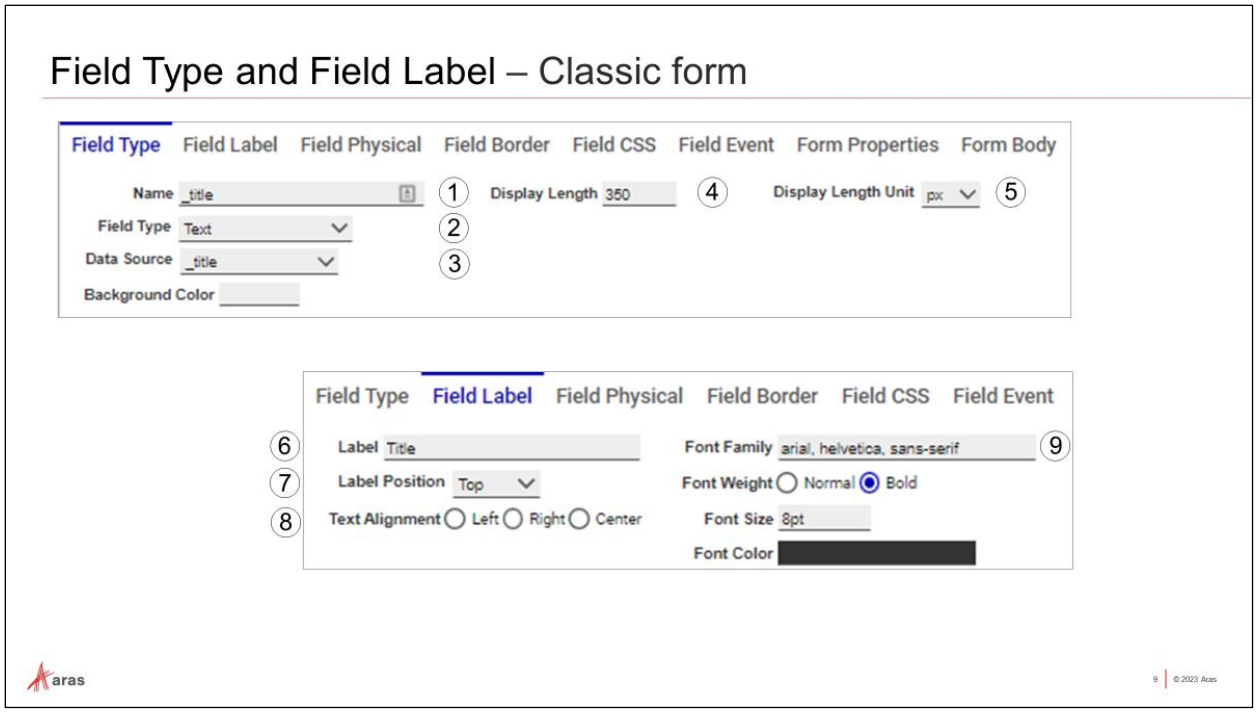
1. While in Edit mode on the Design Request Form, select the following properties from the Unused Properties drop-down and adjust their settings:

Field	Label	Tab Stop	Tab Index
classification	Classification	Checked	110
state	State	Unchecked	
owned_by_id	Request Owner	Checked	20

The Form should look similar to this:



2. Click Done.



Field Type and Field Label – Classic Form

Field Type and Labels are defined using the following fields:

① **Name**

Field name

② **Field Type**

Text: a simple text box

Password: a text box with masking

Text Area: a text box sized using rows and columns

Checkbox: a checkbox for true/false values

Nested Form: A Form within a Form.

Class Structure: a field that enables the user to classify the item

Group Box: a field to group other fields together.

For Lists

Dropdown: a dropdown box

Radio Button List: a set of radio buttons

List box Single Select: select one

List box Multi Select: Hold Ctrl to select multiple items

Checkbox List: a list selectable as checkboxes

Others

Date: a text box with a calendar control

Item: a text box with an item picker/search dialog

File Item: choose File Item

Less commonly used

Label: a simple text placement

Formatted Text: can contain HTML

Multilingual String: Add a String supporting different Languages

Button: a simple button (needs a Method to act)

HTML: a container for arbitrary HTML

Color: a text box with a color picker control

Color List: Interprets List Value as Background color

Image: allow a selectable image to display on Form

③ **Data Source**

Name of the data source property

④ **Display Length**

Length of Field

⑤ **Display Length Unit**

To support correct field sizing in multiple browsers, you can select the appropriate character measurement unit from the list provided. The following units of measurement are available:

px = pixel

em = character in current font size

% = percentage (1em = 100%)

in = inches

cm = centimeters

mm = millimeters

ex = x-height of current font

pt = point (1pt = 1/72 inch)

pc = pica (1pc = 12pt)

⑥ **Label**

Label text

⑦ **Label Position**

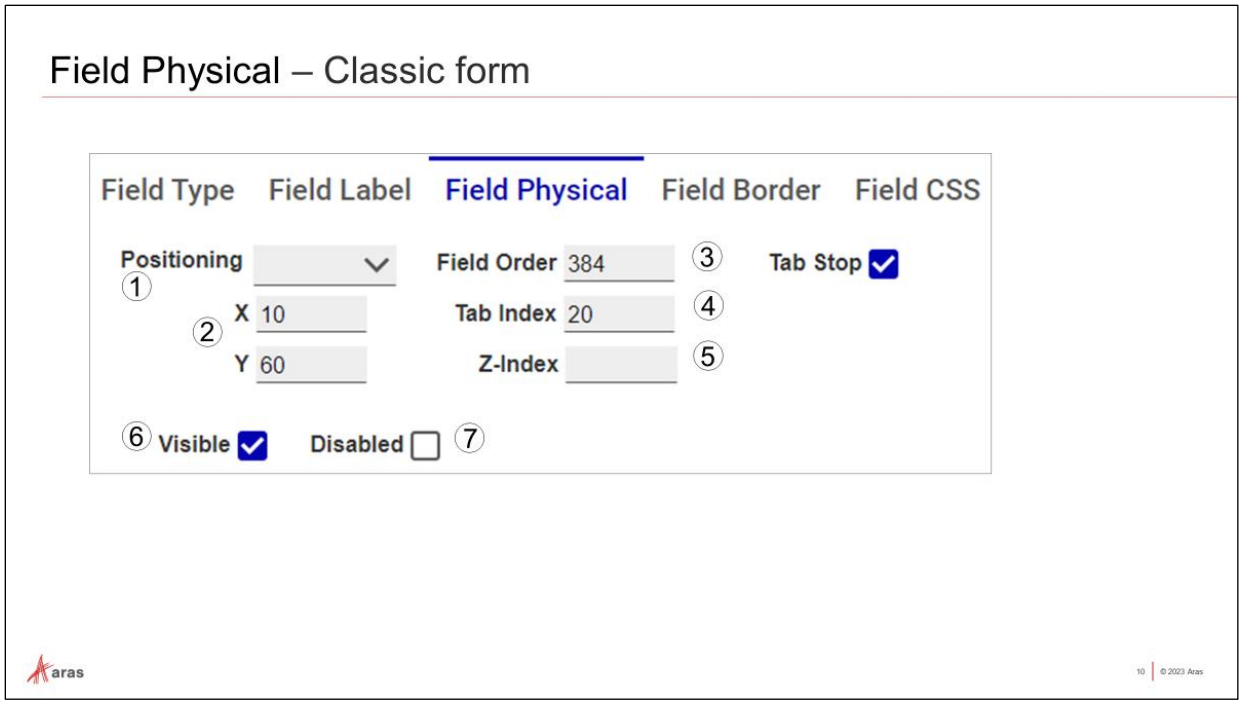
Top, Bottom, Left, Right

⑧ **Text Alignment**

Left, Right, Center

⑨ **Font Information**

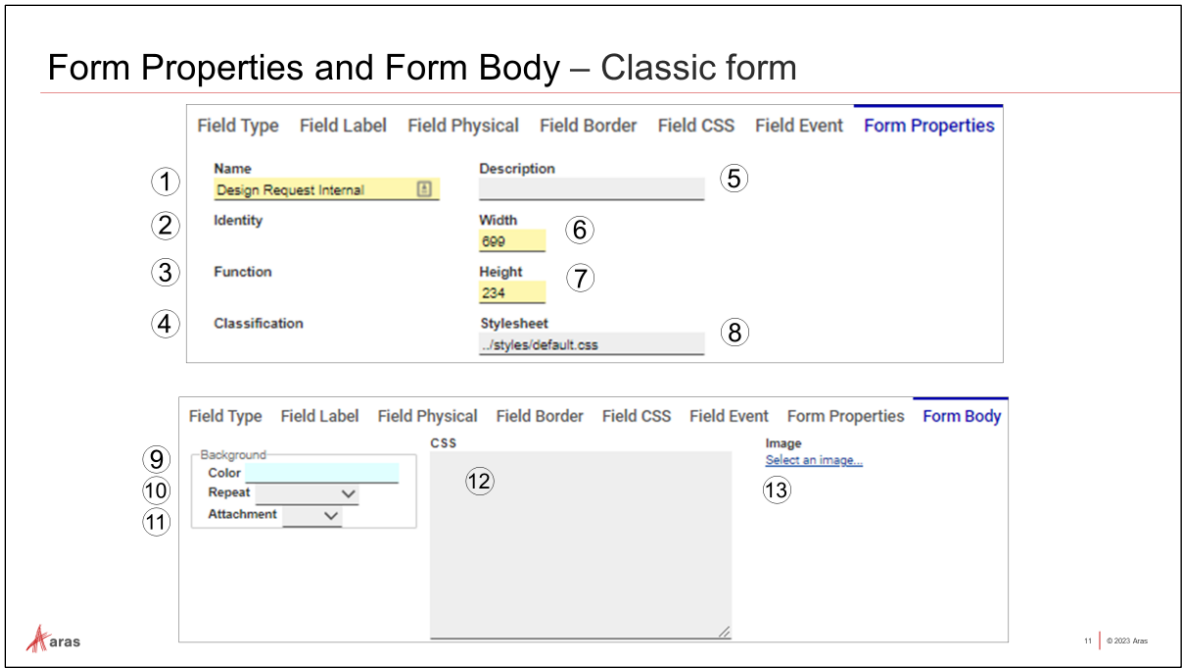
Specific font information about the label for this field



Field Physical – Classic Form

The following fields define a Physical Field:

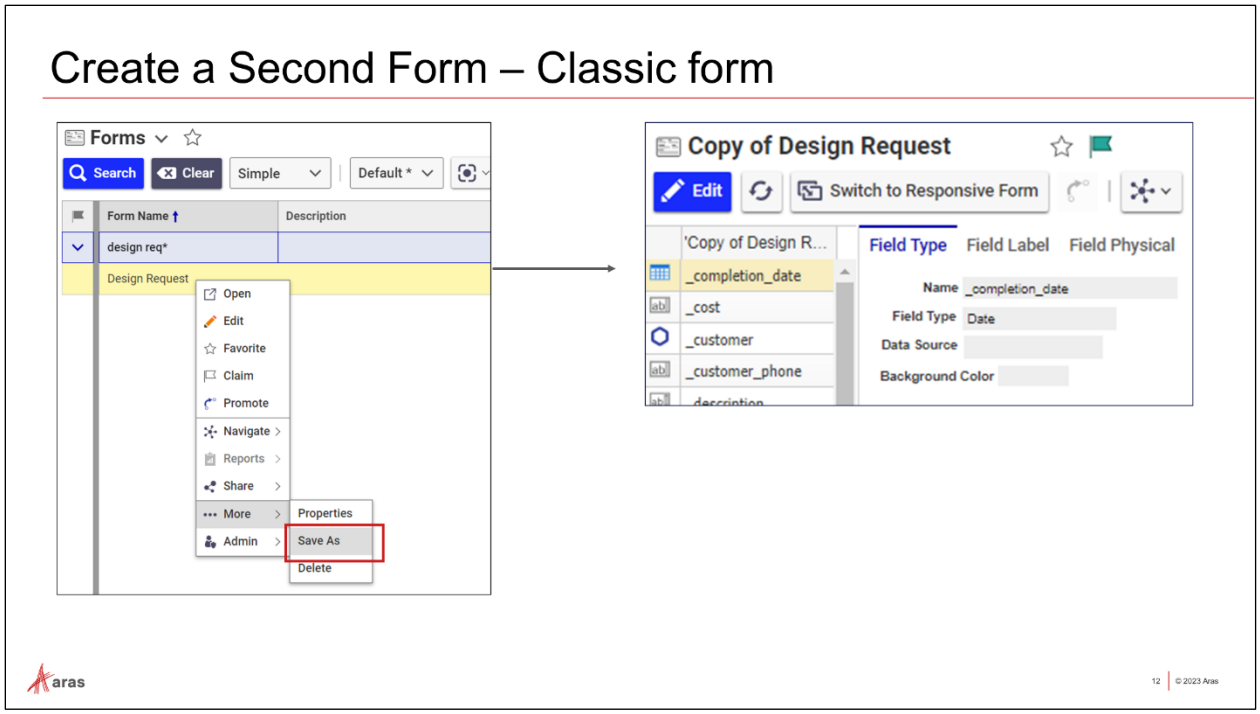
①	**Positioning**	Absolute X and Y: measured from top left of Form, default and recommended. Relative X and Y: measured from where browser would place the field in field order. Static: position determined by browser, X and Y ignored.
②	**X Y**	Row, Column Position. X and Y are measured in pixels.
③	**Field Order**	Number representing the order of each field in a Group Box
④	**Tab Index**	Tab key order of field if Tab Stop checkbox is selected.
⑤	**Z Index**	Order of layered items (when fields overlap). Fields with higher Z-Index are drawn on top of Fields with lower Z-Index.
⑥	**Visible**	Suppress field from view when unchecked. All fields added to a Form are Visible by default.
⑦	**Disabled**	Field is read only when checked.



Form Properties and Form Body – Classic Form

Form Properties and Form Body are defined using the following fields:

①	**Name**	Name of the Form.
②	**Identity**	Defines who has access to this Form (defined in ItemType View).
③	**Function**	Specifies what the Form is used for (editing, viewing, printing, etc.) Default includes edit and view.
④	**Classification**	ItemType classification associated with this Form.
⑤	**Description**	Details about the Form.
⑥	**Width**	Sets the actual width of the Form (in pixels). Impacts the size of any Form displayed in a tab.
⑦	**Height**	Determines the actual height of the Form (in pixels). Impacts the height of an accordion (if set too small, fields set may not be visible and a vertical scrollbar is added to the accordion).
⑧	**Stylesheet**	Stylesheet that is applied to the Form when opened.
⑨	**Background Color**	Background Form color.
⑩	**Background Repeat**	Repeat Form background image
⑪	**Background Attachment**	Scroll – moves with scroll bar – Fixed – no scrolling
⑫	**CSS**	Apply custom CSS elements.
⑬	**Image**	Background Image.



Create a Second Form – Classic Form

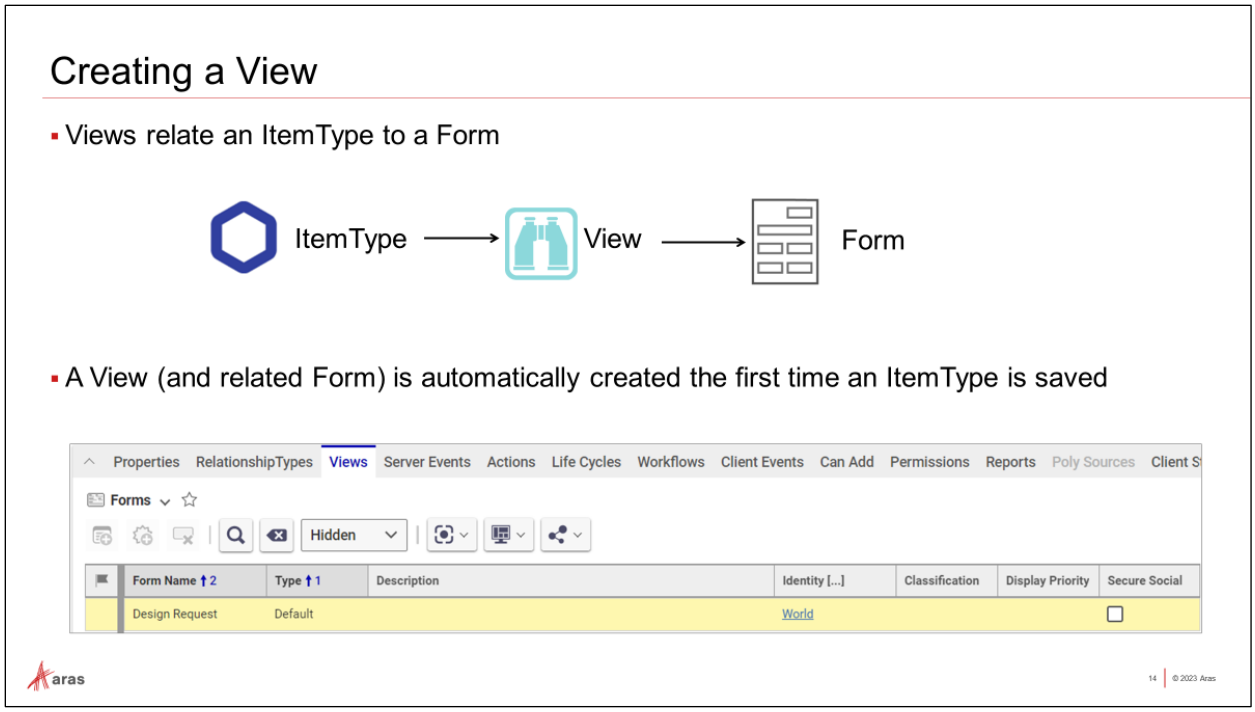
The second classic Form will display only when printing an Item (Share button > Export to PDF).

Note: a newly copied Form is not yet linked to the Itemtype like the original was. It must next be added to the Views tab in the target Itemtype. Make sure to close the Form to clear cache. Re-open it from Views once added and saved.

Try it … Copy a Form

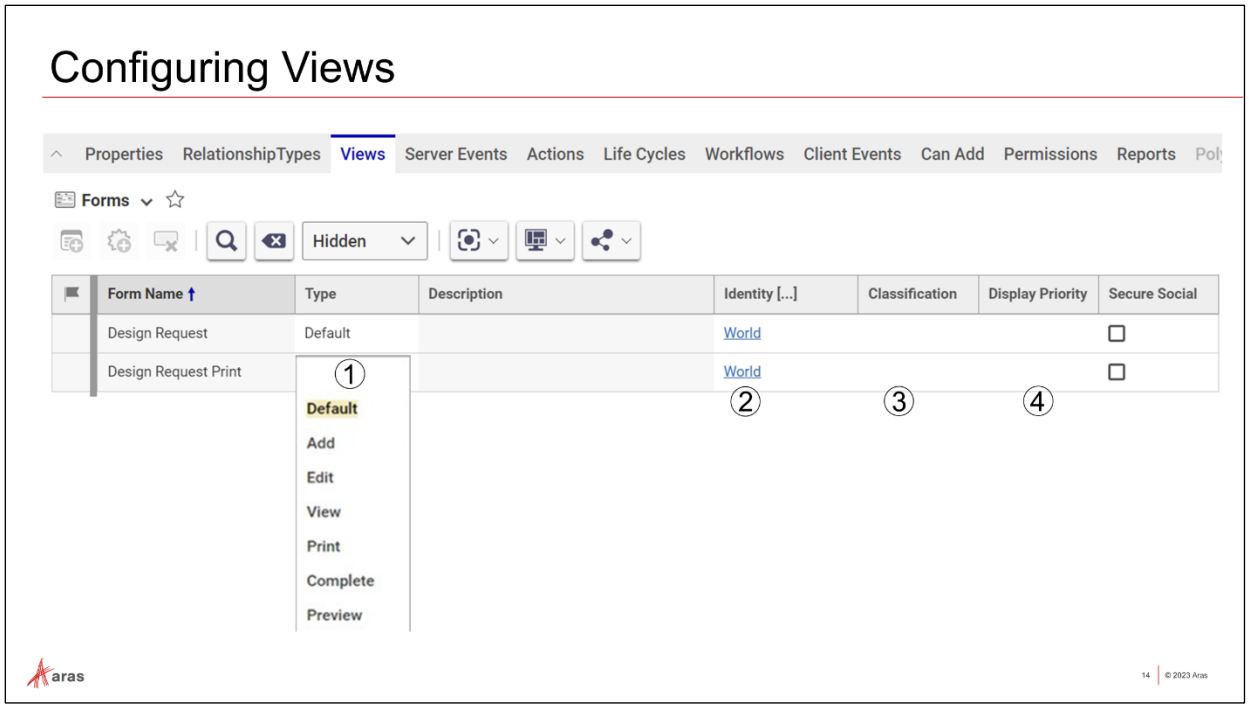
We will use the Forms search grid to copy our existing form.

1. Navigate to Administration > Forms, open the Forms search grid, and search for the **Design Request** Form.
2. Right-click on the **Design Request** Form, and choose More > Save As. Notice that the configuration of fields from source Form were copied in the new Form's classic definition.
3. Change the Name to *Design Request Print*.
4. Click Done. Unclaim the Form.
5. Close the Form tab.



Creating a View

Views are the relationship between an ItemType and a Form. For an end user to see the Form, you must associate it with an ItemType. Aras Innovator automatically creates a view to a Form with the same name as the ItemType the first time you save an ItemType to the database. Additional Forms can be associated with an ItemType using the Views relationship tab.



Configuring Views

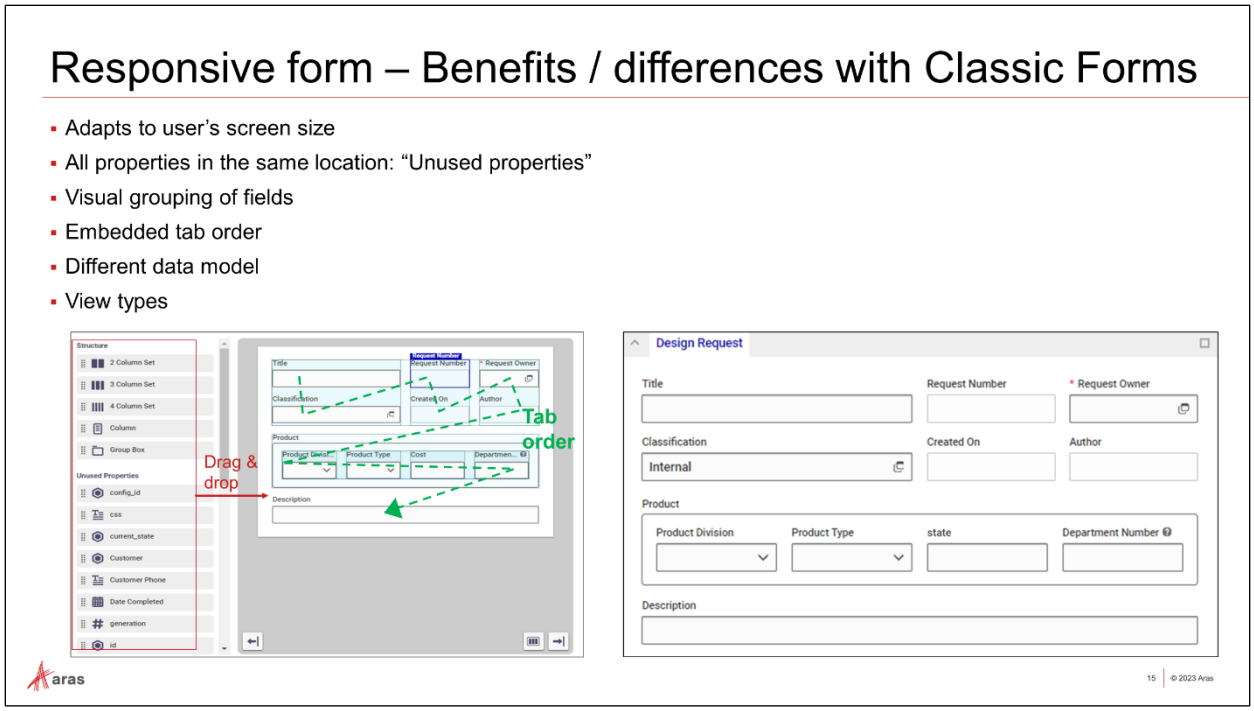
Multiple views can be configured for an ItemType to provide custom displays based on function, identity, and classification. There must always be one Form that is not classified, with its Type set to either Default or Add.

①	**Type**	Default – Form used when adding, editing, viewing, or printing an Item. Add – Form used the first time an Item is added. Edit – Form used when the Item is locked for editing. View– Form used for viewing the Item. Print – Form used to print the Item. Complete – Form used for configurable InBasket dialog. Preview – Display when Preview > Form is enabled in the Search tab.
②	**Identity**	Form available for members of defined Identity.
③	**Classification**	If Class Structure is implemented, the Form to display for a specific class.
④	**Display Priority**	If more than one Form meets the selection criteria, the Form display order priority (1 = highest priority, 2, 3, etc.).

Try it … Create and Classify a View

In this example, we add a Print View type and allow only employees to use this view.

1. Open the **Design Request** ItemType for Editing. On the Views tab, click the Add Forms button.
2. Search for and select the **Design Request Print** Form.
3. Set the Type to **Print**, and the Identity to **All Employees.**
4. Click Done.
5. Open the **Design Request Print** Form, make two fields invisible (e.g., Description and Date Completed). If needed, reposition the bottom-left fields to another location. Click Done.



Responsive form – Benefits and differences with Classic Forms

Adaptation to user' screen size: This Form is responsive: responsive Design is a general design principle that describes a user interface that adapts its layout and dimensions to various devices or screen sizes. Responsive Forms use a container-based approach to define the layout of a Form. This means that content can reflow to use the browser window's space more effectively – minimizing the need for scroll bars and reducing unnecessary whitespace.

On the contrary, Classic Forms render content using a static layout with X & Y position of each field, that doesn't change based on the size of the user's browser window. Users will be presented with scroll bars when a Classic Form's static content does not fit in frame, or they will see additional whitespace when the static content occupies less space than the current frame.

All properties in same location: both hidden and unhidden properties appear in the same location for drag-and-drop, improving the user-friendliness.

Easier visual grouping of fields: the column sets and group boxes enable a clearer grouping of fields. The automatic alignment of those fields also makes the UI easier to read.

Tab order: related to the navigation across fields using Tab key. For responsive Forms, this setting is embedded in the responsive Form, to reduce the efforts for the administrator.

The tab order is setup automatically by the system according to the fields position on the Form:

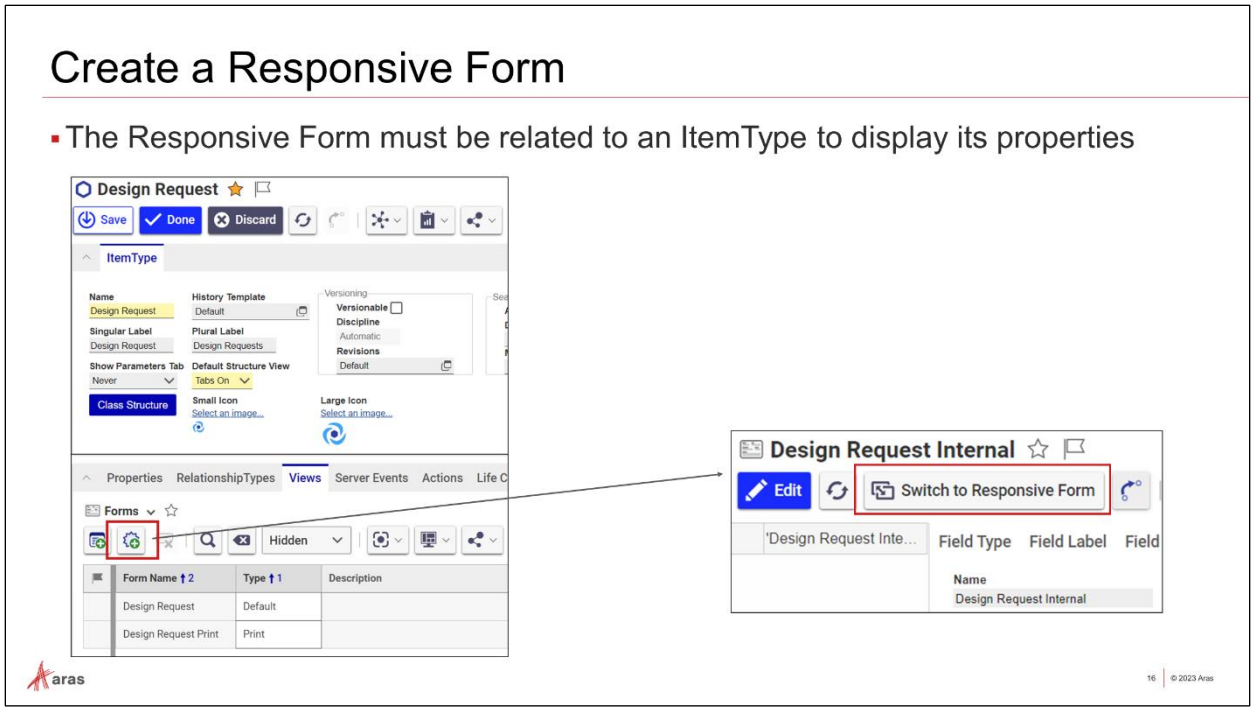
- From top-left to bottom-right corner
- First on fields within one column
    - Then, on columns within a column set/group box
        - Then, moving from one column set/group box to another (note: column sets/group boxes can only be stacked on top of each other, not side-by-side)

Data model: Both Classic and Responsive Forms use the Form ItemType to define Forms for the Aras Innovator client. However, each approach uses a different data structure under the Form item to describe the layout, behavior, and content of the rendered Form. This parallel architecture enables administrators to adopt Responsive Forms with minimal disruption to packaging and deployment. It also enables customers to continue using their existing Classic Forms.

Lists : for lists, List non blank setting does not exist on Responsive Forms. You can use the Required setting on the list property on related item type.

View types: for the moment, only Default view type is enabled for Responsive Forms. If you need to use other types (print, edit, preview, …), use classic Forms instead.

A single Form item might have both Classic and Responsive definitions present at the same time. However, Aras Innovator will only display one of them, according to the active definition. You can switch from one definition to the other at any time. Modifying the layout, behavior, or content of a Classic definition does not affect the Responsive definition, and vice versa. For example, after form creation, if you do modifications in the classic form then switch to responsive form, the responsive definition will be empty.



Create a Responsive Form

The Responsive Form must be related to an ItemType to display all the ItemType properties as Unused properties for selection to build the Form.

Note: currently, responsive Forms only support the Default view type. For view types like Add, Edit, View, Print, Complete, or Preview, users are advised to use classic Forms.

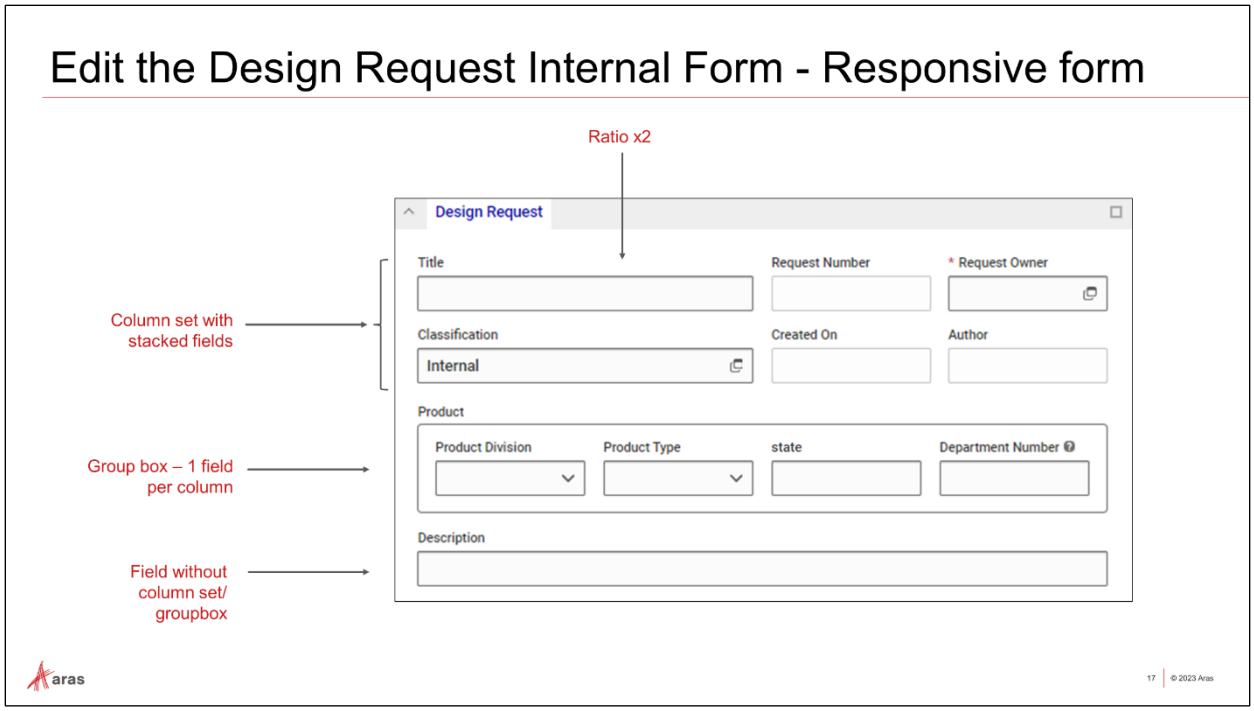
Try it … Create a Responsive Form

The responsive form will be used for the Internal classification. After creation, we will switch to Responsive Form. We also drag and drop several fields including the Department Number field. We will not display Customer Name and Phone fields, which only appear in the classic default Form.

1. Open the **Design Request** ItemType for editing. Open the Views tab.
2. Click New Form.
3. Provide the following information in the Views tab:

Form Name	Type	Identity	Classification
Design Request Internal	Default	World	Internal

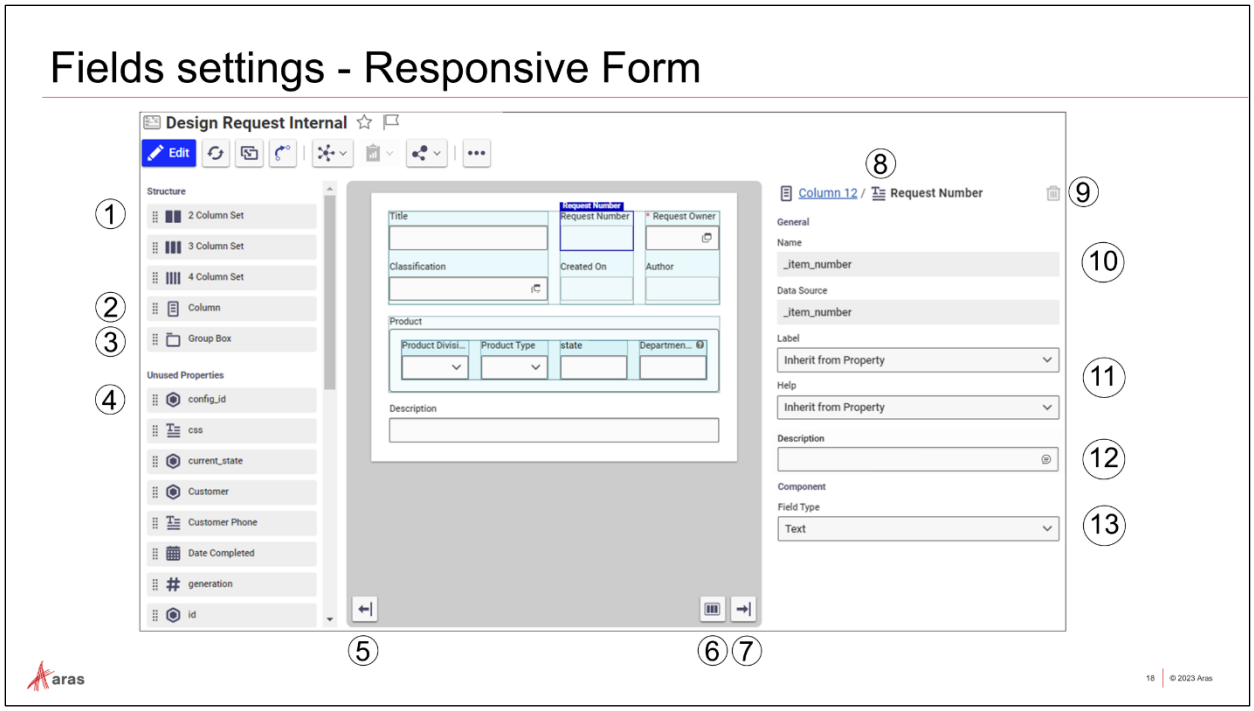
4. Click Done.
5. Right-click the **Design Request Internal** Form and open it.
6. On the Form, click ‘Switch to Responsive form’. Leave the tab open for the next try-it.



Edit the Design Request Internal Form – Responsive Form

Our responsive Form will contain different Aras responsive Form features:

- Column set: container holding multiple columns, each column can contain one of several fields.
- Group box: a column set with label and border displayed by default.
- Individual field dragged-and-dropped directly on the Form.
- Column with double width compared to other columns.



Fields settings – Responsive Form

Responsive Forms use an enhanced admin user interface allowing admins to define a container-based layout by dragging and dropping structural elements and fields into the Form. Administrators can also configure the layout behavior of the Form and preview how it will look and behave for end users.

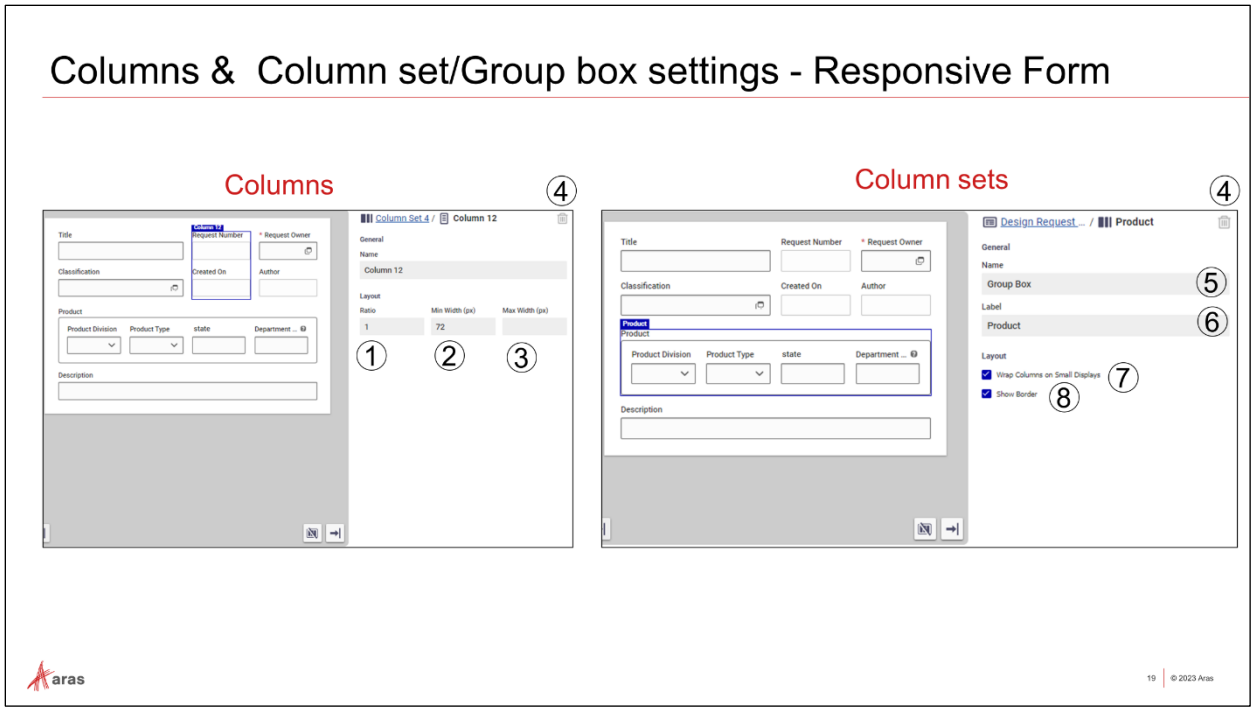
Note: drag and drop on the white part of the middle canvas, not the grey part.

①	**Column set**	Container holding multiple columns. Each column can contain one of several fields. Column sets help organize and arrange Form fields or content side by side on wider screens. Users can drag-and-drop column sets with 2, 3 or 4 columns. Label and border are hidden by default.
②	**Column**	When dragging-and-dropping a column in a column set/group box, an additional column is added. Columns width get resized accordingly to keep the same total width. Admins can add as many columns as needed, the only limit is the minimum column width.
③	**Group Box**	A column set with label and border displayed by default.
④	**Unused Properties**	All properties from the related ItemType, both hidden or unhidden. Users drag and drop properties within columns or directly on the canvas. Reminder: if the responsive Form is not linked to any specific ItemType, then the Unused Properties will not appear.
⑤	**Close Panel**	Closes the left-hand side Element panel.
⑥	**Hide Guides**	Displays the Form's actual appearance as it will be shown to the user
⑦	**Close Panel**	Closes the right-hand side Detailed panel.
⑧	**Hierarchy**	Allows to navigate in the Form > Column sets > Columns > Fields hierarchy.

⑨	**Delete**	Deletes the selected element.
⑩	**Name**	Name of the field displayed on the designer. Inherited from ItemType property.
⑪	**Label & Help**	Label stands for the Field label displayed to the end users. Help is related to tooltip. For both, the possible values: - Inherit from property: shows the label/help from the ItemType property - Hide: label/tooltip is hidden for users
⑫	**Description**	A second configurable "label" beneath the input field, which administrators may use as descriptive helper text to assist end-users with understanding what data should be input to the field or how it relates to business processes.
⑬	**Field Type**	Shows the field type based on the property data type (e.g. Text, Combobox, Number, Item, Checkbox, Image). Only Text property type has multiple field types as follows: • Text: A concise input field suitable for short entries. • Text Area: A larger input space ideal for longer pieces of text or paragraphs.

Try it … Add Properties on the canvas

1. Open the **Design Request Internal** Form for editing.
2. Drag and drop a 3 column set on the white area of the canvas.
3. Drag and drop the following properties:
   a. **Title** and **Classification** in the first column.
   b. **Request Number** and **Created On** in the second column.
   c. **Request Owner** and **Author** in the third column.
4. Drag and drop a Group box on the white area of the canvas, below the column set. Notice that one column appears by default in the group box.
5. Click on the Structure Element *Column* and drag and drop 3 additional columns inside this group box (4 columns in total).
6. Drag and drop one property in each column, in this order: **Product Division**, **Product Type**, **state**, **Department Number**.
7. Select the Product Type field, then fill the Description as follows: *Product types values depend on the Product Division value selected*.
8. Drag and drop the *Description* property at the bottom of the canvas' white area. Notice that this field can have field type Text or Text Area.
9. Click Save.



Columns and Column set/Group box settings - Responsive Form

The columns, column sets and group boxes can be setup to configure the labels, behavior of those elements when the screen gets reduced, etc.

You can use the top right corner to navigate within the hierarchy of Responsive Form > Column set/Group box > Column > Field.

Column settings

①	**Ratio**	Space each column occupies relative to others in terms of width. For instance, a column ratio of "2:1" between two columns means that one column is twice as wide as the others.
②	**Min Width**	Minimum width of the column in pixels, used when screen is reduced horizontally.
③	**Max Width**	Maximum width of the column in pixels.
④	**Delete**	Deletes the element (here, a column).

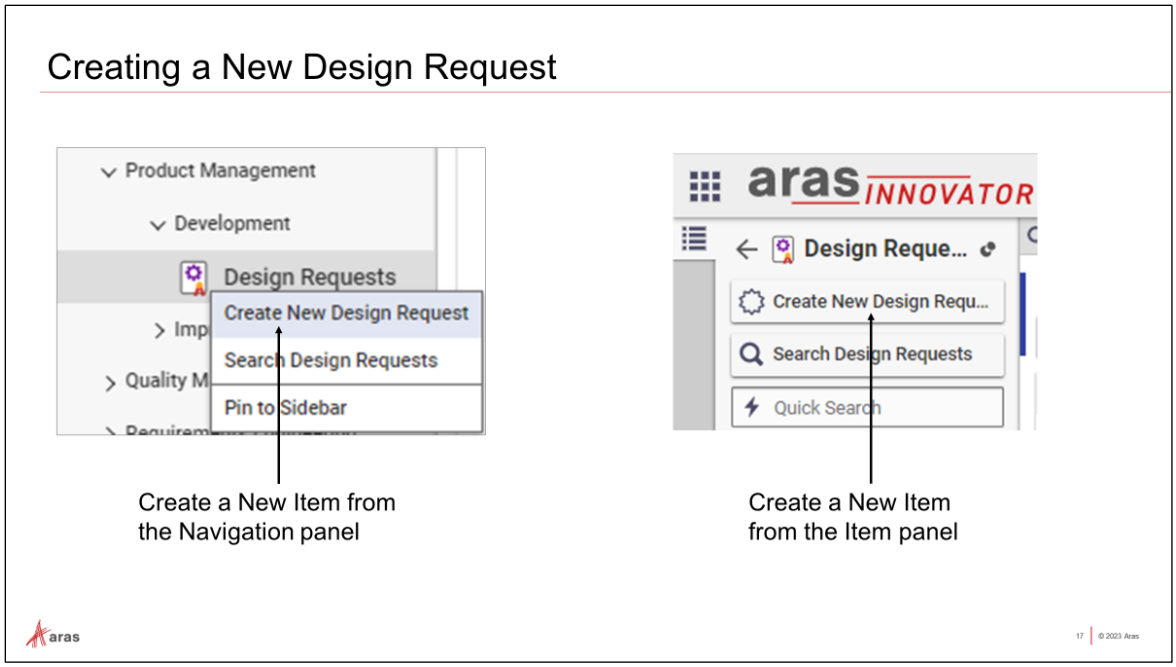
Column set & Group box settings

④	**Delete**	Deletes the element (here, a column set / group box).
⑤	**Name**	Provides name of the element.
⑥	**Label**	Editable name of the element - is displayed to the user.

⑦	**Wrap Columns on Small Display**	Adjusts columns to stack vertically (or "wrap") instead of aligning horizontally when viewed on devices with limited horizontal space. This ensures content remains legible and user-friendly regardless of the screen size.
⑧	**Show Border**	Displays the border of the column set.

Try it … Change the Columns and Group box settings

1. Select the **Title** field of the first column.
2. On the Detailed panel (right-hand side), click on the Column link. This opens the column's detailed panel.
3. Change the Ratio to 2 for this first column.
4. Notice the min and max width settings ; keep their default values.
5. On the Detailed panel (right-hand side), click on the Column Set link. This opens the Column Set's detailed panel. Notice the column set settings, including *Show Border* which is unchecked by default.
6. Click on the Group box header, then change the Label to **Product**.
7. At the bottom, click Close Panel on both left and right sides.
8. Click Hide Guides at the bottom.
9. Click Done.



Create a New Design Request

Now that the minimum requirements have been defined and a class structure established, you can test to see if you can create a new Design Request from the TOC Pane.

Create a new Design Request Item by either right-clicking on the Design Requests ItemType button in the TOC and selecting Create New Design Request, or by left-clicking on the Design Requests ItemType button in the TOC and then clicking the Create New Design Request button.

The **Design Request Form** should allow for the following behaviors:

- The customer field should allow the selection of a Customer Item and the customer's phone number should display in the Customer Phone field after saving the Design Request.
- When a user selects a Product Division, the Product Type list is filtered.
- An automatically generated Sequence number displays upon Save.
- The user should be able to select an Image to display as a thumbnail on the Form.

Try it … Create a Design Request

1. Navigate to Product Management > Development > Design Requests and create a new Design Request.
2. If necessary, maximize the Form to view all the fields.
3. Leave the **Classification** field as is and enter values into the remaining editable fields to check the new Form.
4. Click the Done button.
5. Click Share > Export to PDF. The properties are displayed with the *Design Request Print* form.
6. Create another Design Request.
7. Enter *Internal* into the **Classification** field and enter applicable values into the remaining editable fields to check the new Form.
8. Notice the content reflow upon changing screen size, tooltip and tab order behaviors.
9. Click the Done button.

Summary

In this unit, you learned how to create an Item View using the Form Editor.

You should now be able to:

- Understand Forms and Views
- Create a User Interface using the Classic Form Editor
- Review Field Settings
- Review Classic Form Settings
- Build an ItemType View
- Build a responsive Form

Review Questions

1. What is the difference between a Form and a View?

2. How many Forms can be assigned to an ItemType?

3. When would you not use RebuildViewAction?

4. How do you control the size of a Form accordion?

5. What are the benefits of Responsive Forms ?

Lab Exercise

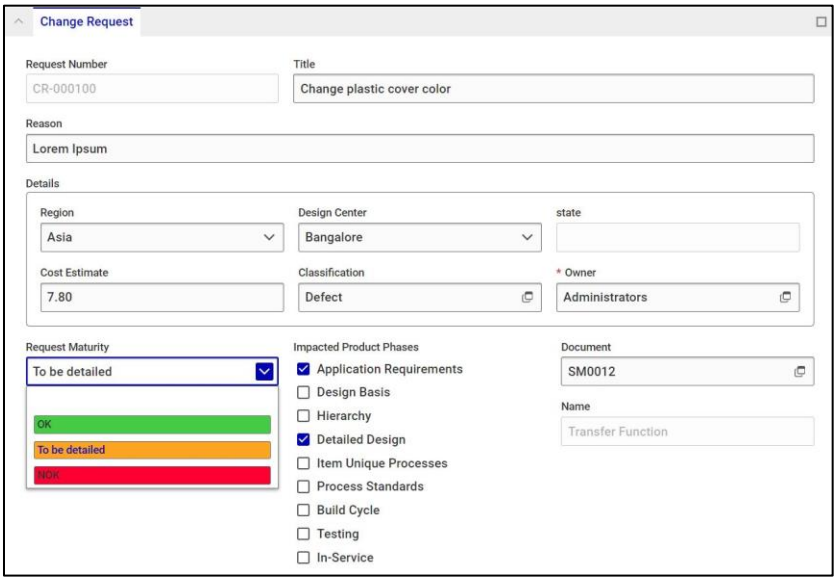
Goal

Configure a Form for an ItemType allowing changes to property values.

Scenario

You edit the automatically generated Change Request Form to allow a user to supply values to the custom properties created earlier. You also add a multi-value list and a color list to this Form.

The result will be the following:



Steps

1. On Administration > List, create a new list:

Name	Description
OK-NOK_Colors	*OK, To be detailed and NOK - Colored list*

2. In the Values tab, add the following values (remark: you can find other color values using an hexadecimal color picker on the Internet):

Label	Value	Sort Order
OK	#45CB45	10
To be detailed	#FDA423	20
NOK	#FF0033	30

3. Click Done and close the list.
4. Open the **Change Request** ItemType and add two new properties:

Name	Label	Data Type	Data source	Length
_maturity	Request Maturity	Color List	*OK-NOK_Colors*	
_product_phases	Impacted Product Phases	Multi Value List	Product Phases	64

5. Select the Views tab in the Relationships Grid.

6. Select the Change Request Form and set the **Identity** to *All Employees*.
7. Save, then right-click on the Change Request Form and select Open from the context menu.
8. Click Switch to Responsive Form.
9. Click the Edit button to edit the Form.
10. Rearrange the custom properties on the Form according to the above screenshot. Hint: the top of the canvas contains:
    a. a 2-columns column set whose second column has a ratio of 2
    b. then a singular field **Reason**
    c. then a group box with 3 columns inside
    d. then a 3-columns column set
11. Test your work by creating a new Change Request Item.
12. Make sure you can select a Region, Design Center and select a related Document Item.
13. When the Item is saved, the name of the selected Document Item should display.
14. Click Done.

This page intentionally left blank.

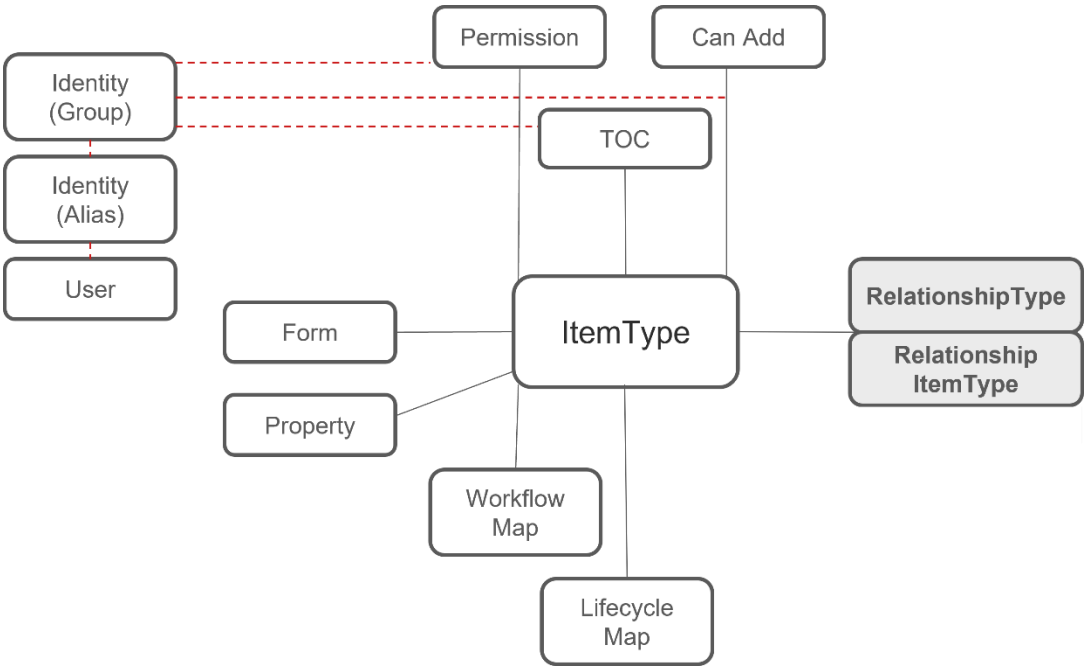
Unit 8 Creating Item Relationships

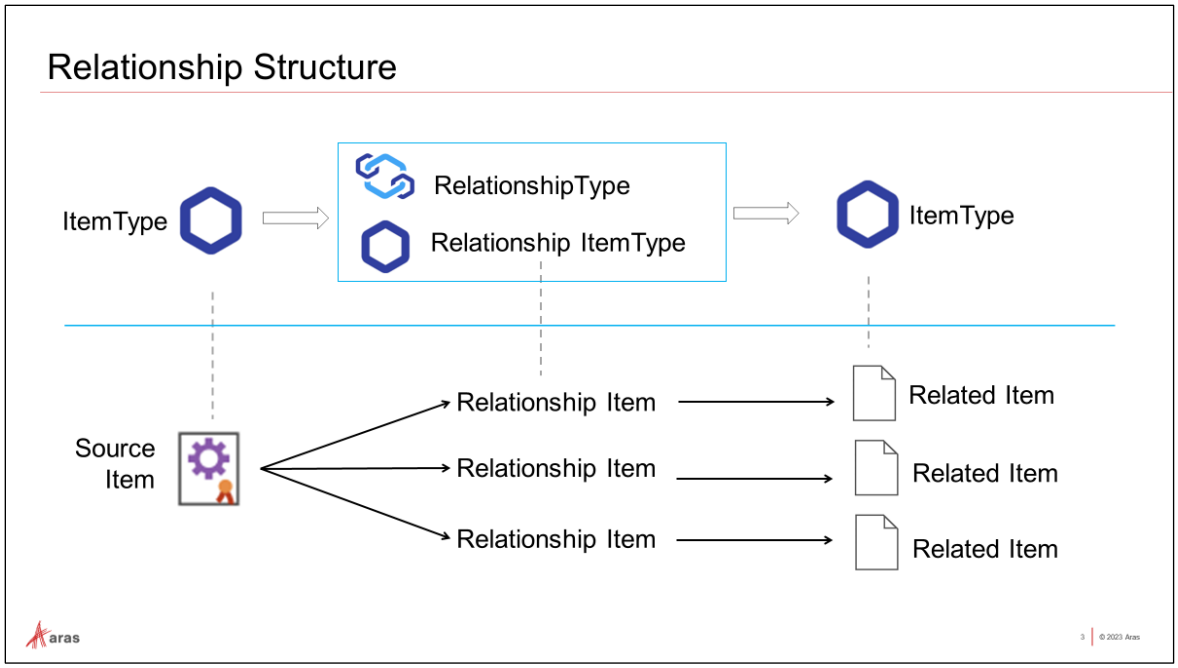
Overview

In this unit, you learn how to build relationships between two ItemTypes. You also learn about typical relationship patterns and how they can be used in different situations.

Objectives

- Define Relationship Types
- Understand the Relationship ItemType
- Review Typical Relationship Patterns
- Explore the Benefits of NULL Relationships





Aras Innovator Relationship Structure

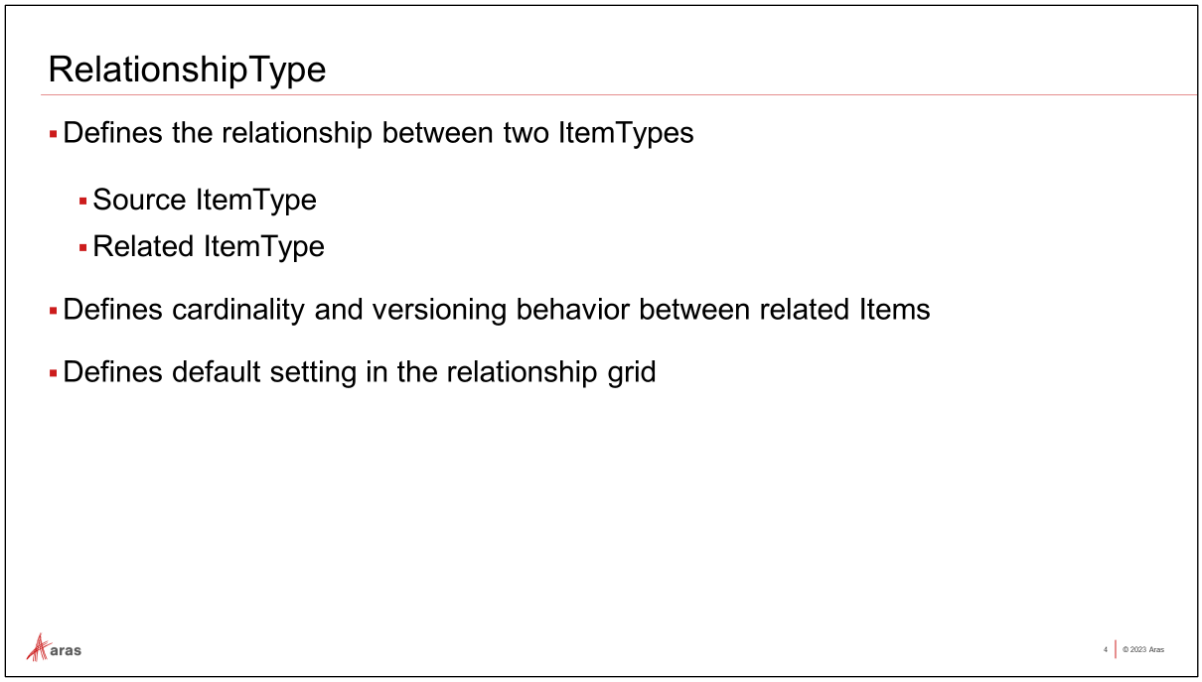
Items may have relationships to other Items using RelationshipTypes. RelationshipTypes determine what ItemTypes instances may be related to other ItemType instances (Items).

When two Items are related in the database, a Relationship Item instance is also created that can hold distinct property values for that specific relationship. For example, in a Part BOM relationship, a value for Quantity or Reference Designator belongs to the Relationship Item between the parent Part and the child Part.

Relationship Items also include the following standard properties:

- related_id: the related (child) Item for the relationship
- source_id: the source (parent) Item for the relationship
- behavior (fixed, float): used for configuration management

Each RelationshipType is represented as a separate tab on the Form. Relationship Items (instances of Relationship ItemTypes) can be added as rows under each tab by users in the Relationships Grid.



RelationshipType

When you create a Relationship, the required source and the related ItemTypes are defined in the RelationshipType Item.

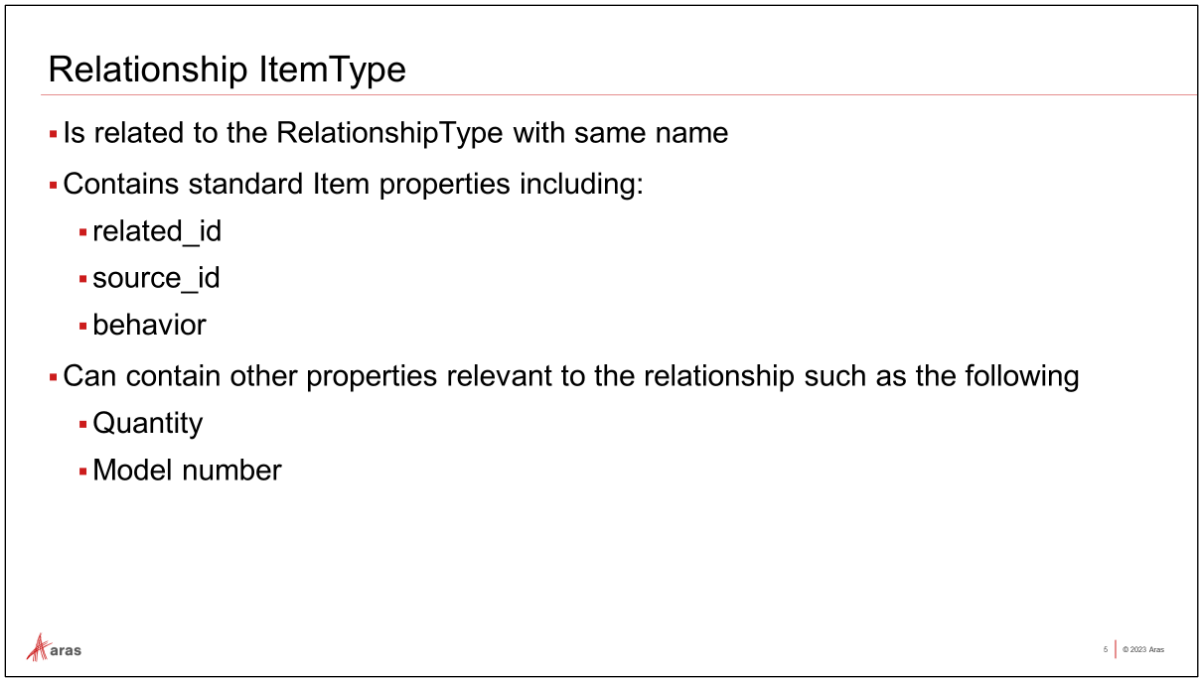
You can configure how many items can be related to the source item (one-to-one, one-to-many) and how versions should be treated among related Items.

You can also configure some general default settings for how items will be displayed on the source item relationship search grid (tab location, auto-search, and several items for page controls).

Additionally, you can configure whether to allow the related items to be selected from items that have been previously created and stored within the database, to allow creation of a new item as a relationship is being made, or both.

Try it … View the RelationshipType Search Grid

1. Navigate to Administration > RelationshipTypes.
2. In ‘Simple’ search mode, enter Part in the Source ItemType [...] column and click the Search button.
3. Note that all eligible Relationships are listed.
4. Enter Document in the Related ItemType [...] column.
5. Note that RelationshipType (the row itself), Relationship ItemType, source and related are all presented in a single view.
6. Experiment with various source/related combinations.

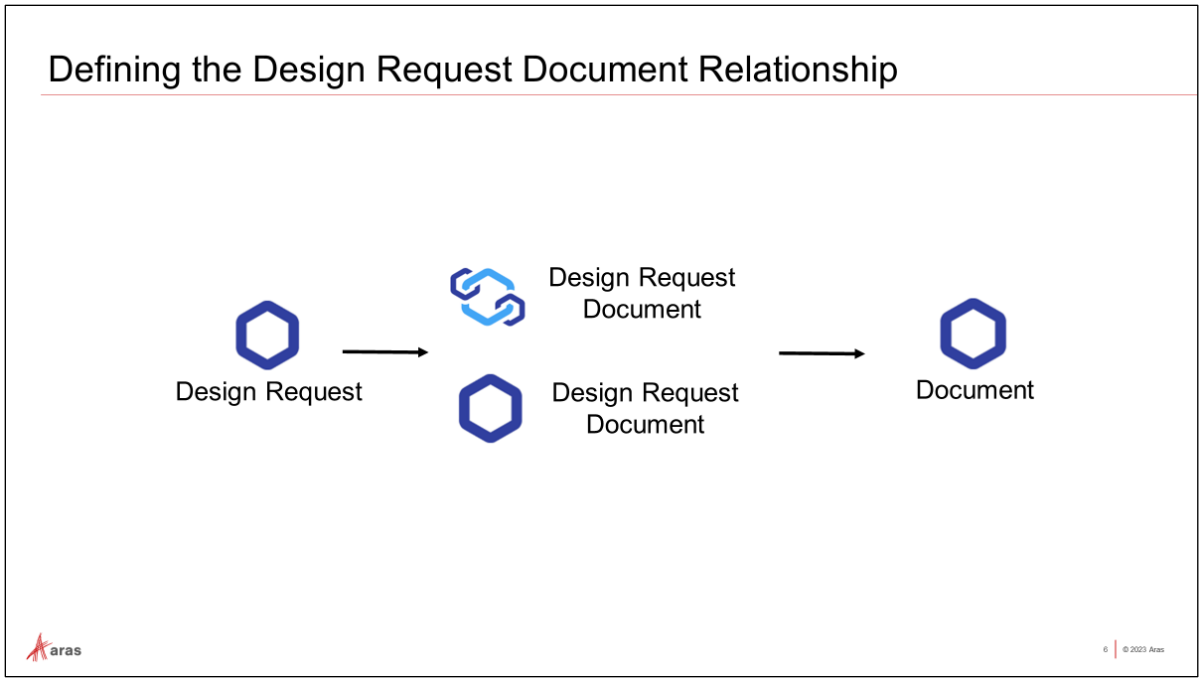


Relationship ItemType

When you create a RelationshipType, a Relationship ItemType is automatically created in the database with the same name. This template defines the information that the Relationship Items will be able to store.

Relationships are also Items, and they can store additional information about the relationship in the database. For example, the Part BOM relationship in the standard PE Solution stores the number of related Parts from a parent Part in a property named Quantity.

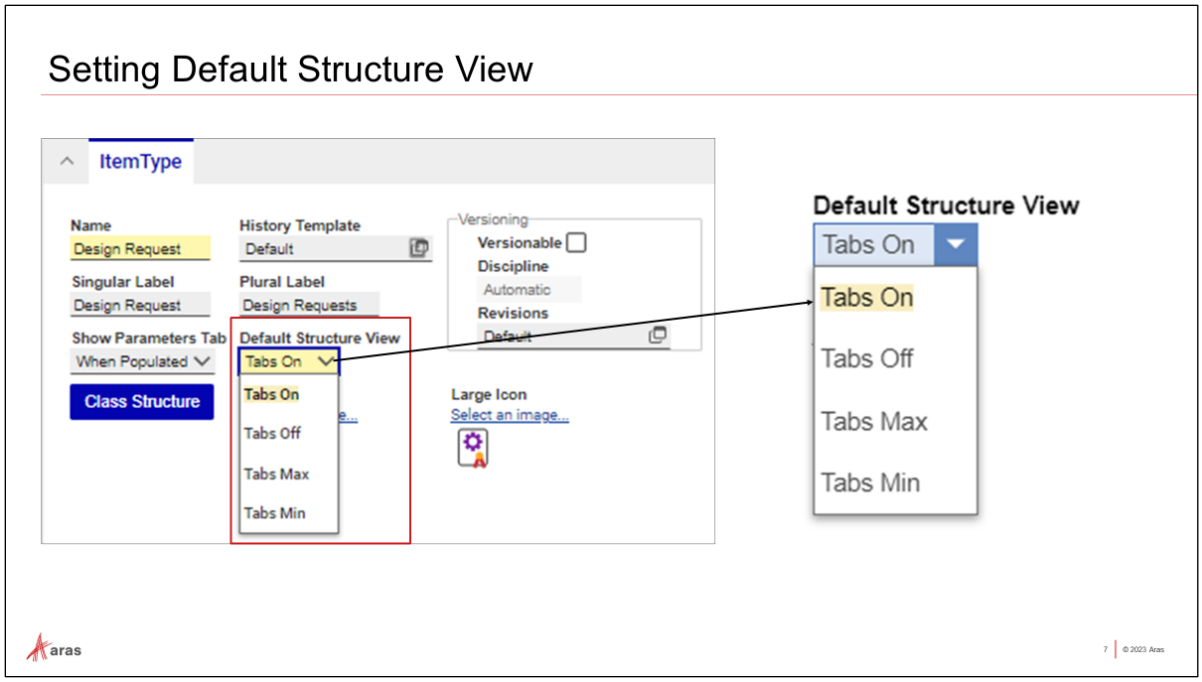
The display of certain properties and the location of columns displayed on the parent item are set on the Property tab of the Relationship ItemType.



Defining the Design Request Document Relationship

One of the requirements for our Design Request solution is to be able to be linked to one or more Document items. Document Items allow physical files to be attached and stored in a vault. We need to create a RelationshipType to define that connection.

In this example, we create a relationship between the Design Request ItemType and the Document ItemType. Note that the Document ItemType already exists in the standard solution database.



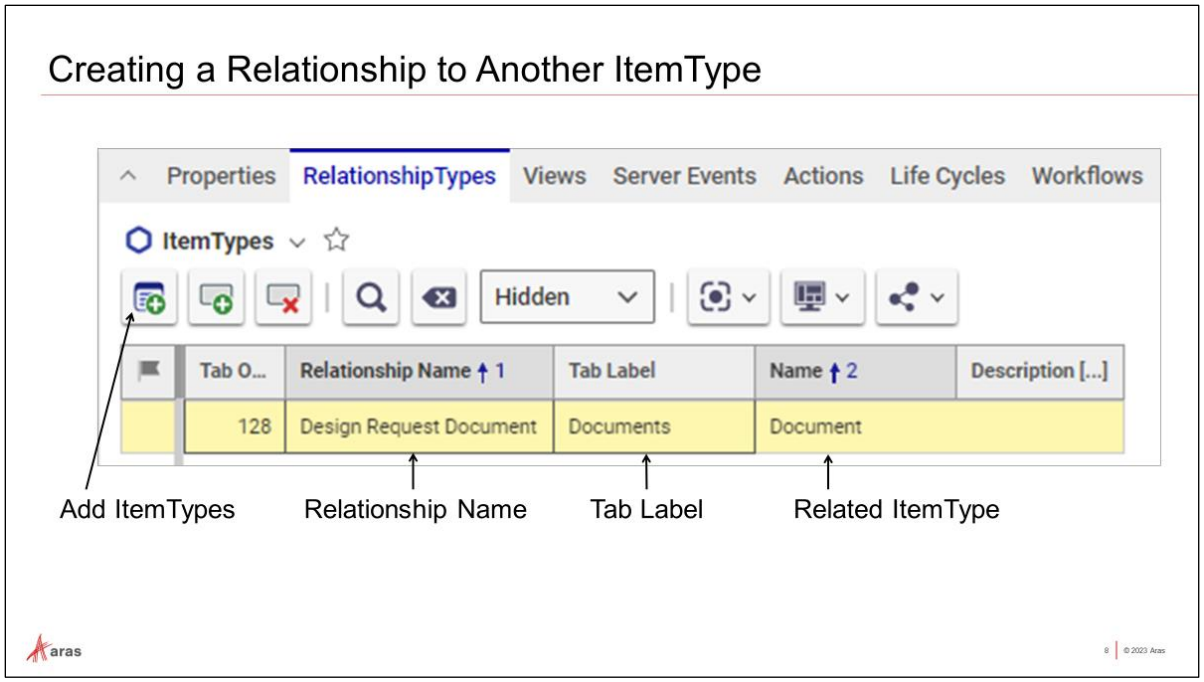
Setting Default Structure View

When a user opens an item containing relationships, the relationships tab grid can be defaulted as shown above. Choose one of the following settings to set the default view of the relationship tabs.

- **Tabs On** – the normal view mode, which displays the relationship tabs in an accordion section.
- **Tabs Off** – the relationship tabs are not displayed in the accordion section (hidden).
- **Tabs Max** – currently, same as Tabs On; reserved for future use.
- **Tabs Min** – the accordion section containing the relationship tabs is collapsed when the Item displays in the window.

Try it … Set Default Structure View to Tabs On

1. Open the **Design Request** ItemType for editing.
2. Use the Default Structure View field's drop-down menu and set it to **Tabs On**.
3. Save the Design Request ItemType.

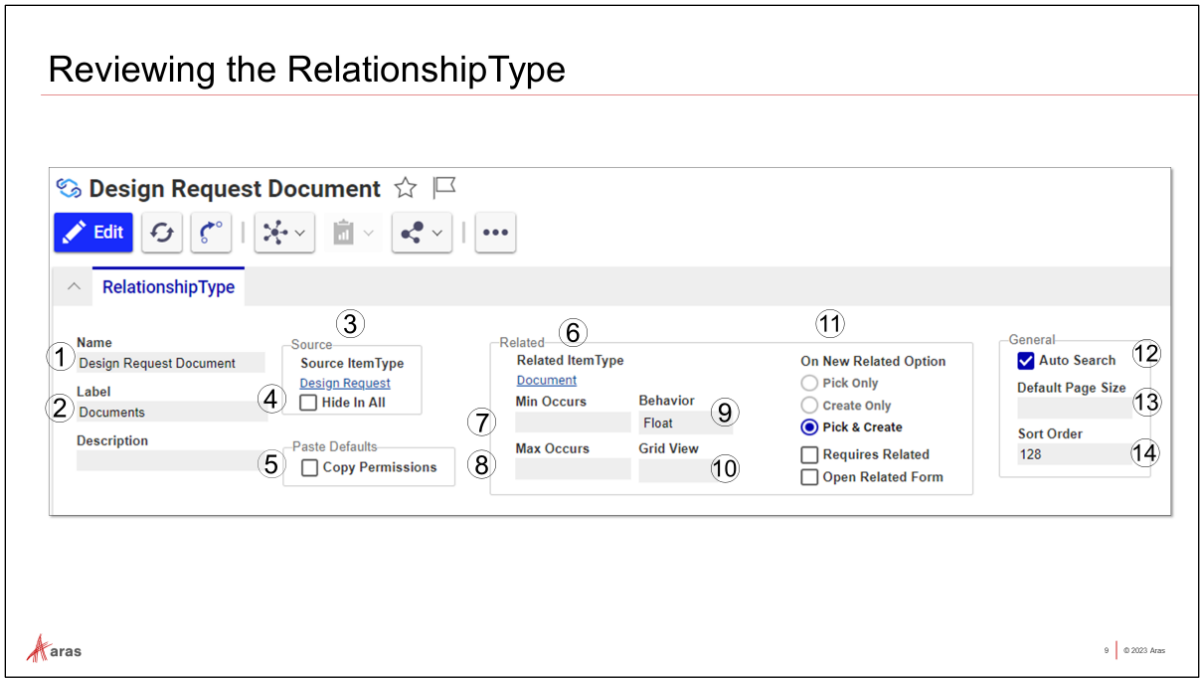


Creating a Relationship to Another ItemType

The recommended way to create a new RelationshipType is by adding a new relationship to a source ItemType. The Name column displays the related ItemType. A relationship to another ItemType can be added using the Add ItemTypes button. The label displays on the tab for the user in the source Item.

Try it … Create a RelationshipType

1. With the Design Request ItemType open for editing, select the RelationshipTypes tab, and click the Add ItemTypes button.
2. In the Select Items - ItemTypes window, choose the **Document** ItemType and click OK.
3. Enter *Design Request Document* for the Relationship Name.
4. Enter *Documents* for the Tab Label.
5. Click Done.

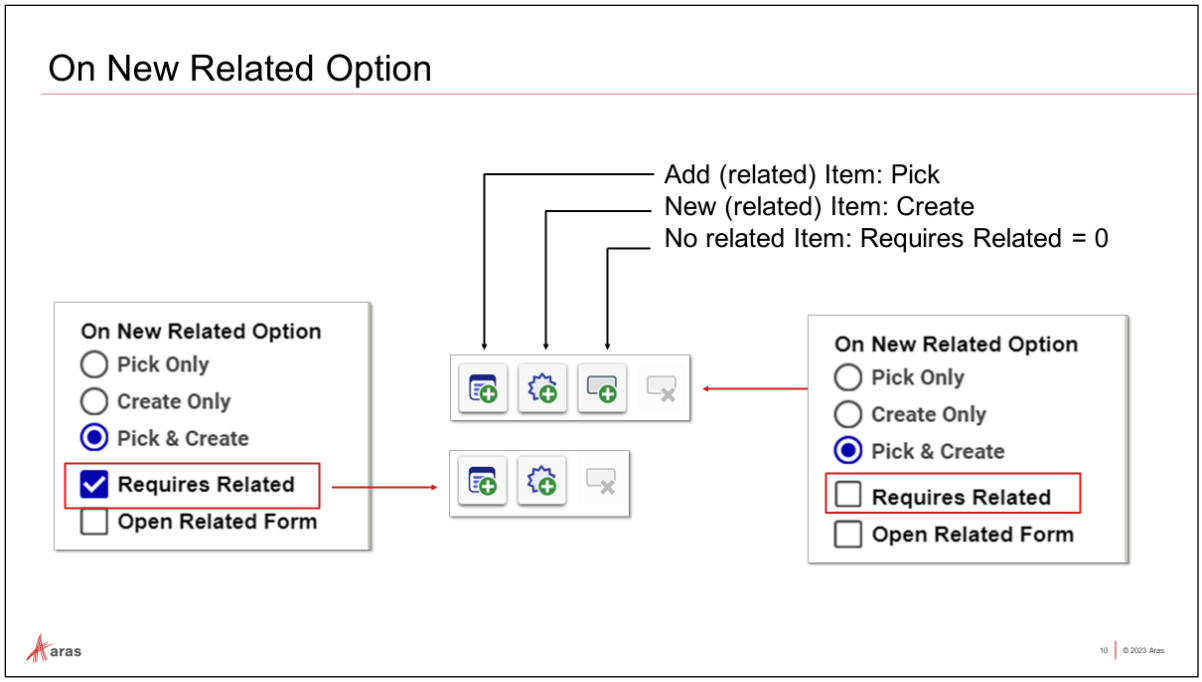


Reviewing the RelationshipType

The following fields define a RelationshipType:

①	**Name**	The Name of the Relationship.
②	**Label**	The Label displayed on the Item Relationships Tab.
③	**Source ItemType**	The Name of the Parent ItemType.
④	**Hide in All**	If checked, hides the relationship on the Relationships Grid of the parent Item.
⑤	**Paste Defaults**	Copy Permissions- a user can copy a relationship from one item to another by choosing the Copy action in the Relationship grid (right click menu). Note: If a user performs a Paste Special and selects the Create Related option, the related Item is copied. If the related Item has a Private Permission, the Private Permission is also copied to the new related Item(s).
⑥	**Related Item Type**	The name of the Child ItemType.
⑦	**Min Occurs**	The Minimum number of related items required.
⑧	**Max Occurs**	The Maximum Number of related items allowed.
⑨	**Behavior**	Establishes whether the links to the related items are fixed to a specific version or float to the most current version.

⑩	**Grid View**	Right - places the related Item Properties (the columns) on the right side of the grid; the relationship properties are listed to the left of those columns. Left - places the related item properties on the left of the grid before all the relationship properties have been listed. Intermix - the columns are ordered only by the sort_order value of each property, so the relationship properties and the related properties will be intermixed.
⑪	**On New Related Option**	Allows the user to select existing Items, create new ones, or both.
	Requires Related	When selected, removes the “No Related” choice in the relationships grid. A user must either select an existing related Item or create a new one.
	Open Related Form	If checked, the related Item form displays, allowing a user to enter data in a Form rather than using the Relationship Grid columns.
⑫	**Auto Search**	Display search results automatically for related Items of this Relationship Type.
⑬	**Default Page Size**	The number of items returned in the Search grid (Results per page).
⑭	**Sort Order**	The Order of the Relationships Tab on an Item.



On New Related Option

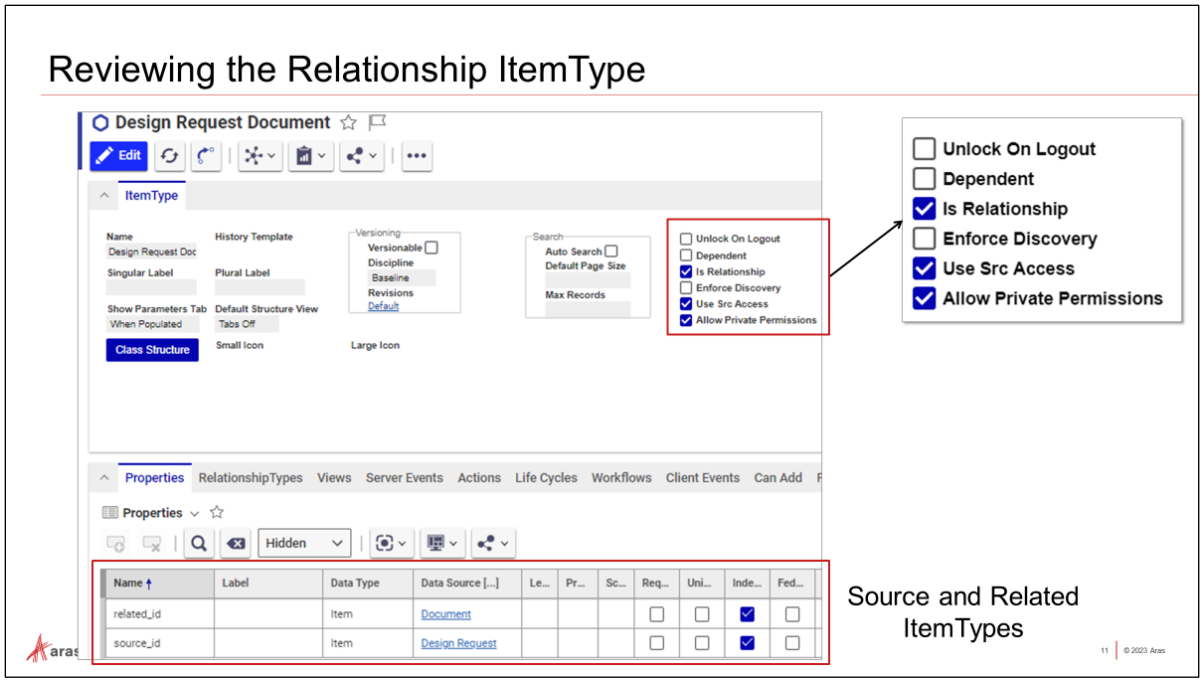
The Aras administrator can configure how users are allowed to add related items in the Relationship Grid. The options are set on the form of the RelationshipType.

- **Pick Only** means the Item must already exist in the database and users can search and choose it. Selecting this option hides the New Item button in the Relationship Grid.
- **Create Only** means users can only create a new Item, they cannot choose an Item that already exists in the database. Selecting this option hides the Add Item button in the Relationship Grid.
- **Pick & Create** allows both options. Selecting this option makes both the Add Item and the New Item buttons visible in the Relationship Grid.
- **Requires Related** means a Related Item must either be chosen or created. Selecting this option hides the button which allows a NULL relationship. If Requires Related is not selected, users can add a new blank row in the Relationship Grid as a placeholder.
- **Open Related Form**: consider selecting it whenever creation of the related item is allowed, and a form exists for its ItemType. This automatically opens the form of the related item for editing.

For our example, we want to allow users to both choose an existing and create a new Document Item. We do not want to allow them to add a placeholder row.

Try it…Require a Related Item

1. Navigate to RelationshipTypes and open the **Design Request Document** RelationshipType for editing.
2. In the *Related* section, select **Requires Related** and **Open Related Form**.
3. Ensure **Pick & Create** is selected.
4. Click Done.



Reviewing the Relationship ItemType

The Relationship ItemType is automatically created and uses the same name as the RelationshipType. You can add properties and behavior to this ItemType if necessary, to supplement characteristics of the relationship.

Relationship Type Options

“Is Relationship” is automatically selected indicating this is a Relationship ItemType.

“Use Src Access” is automatically selected indicating this Relationship ItemType will use the same security permissions as the parent ItemType (in this example “Design Request”). If you uncheck this option, you must provide the “Can Add” Identities as well as the default Permission for this ItemType.

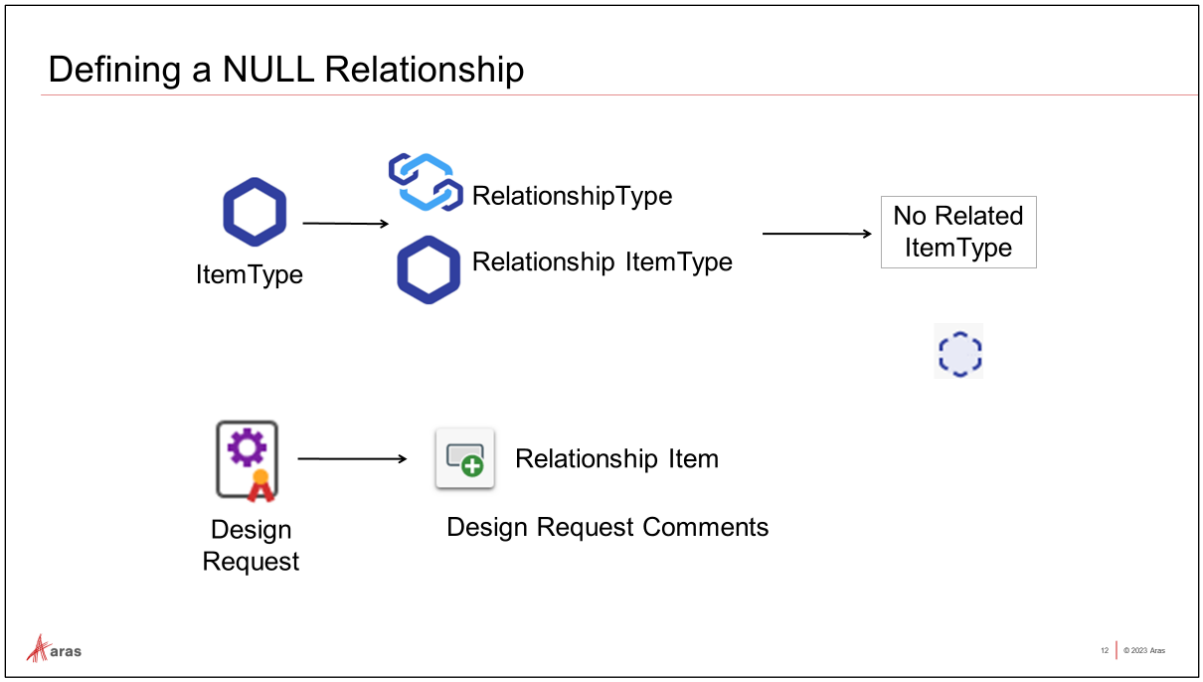
Dependent Option

The Dependent option is useful to prevent orphan Items in the database. If an Item cannot exist on its own (must have a parent Item), select this option on the related ItemType. This prevents a user from creating an Item of this type unless it is part of a relationship. If the parent Item is deleted, the dependent Item is deleted as well (a cascading delete).

Example: A Part Price Quotation cannot exist without a corresponding Part. The Price Quotation ItemType is marked as Dependent, which means that quotes can only be created on an existing Part.

Try it … Create a Design Request Document Relationship

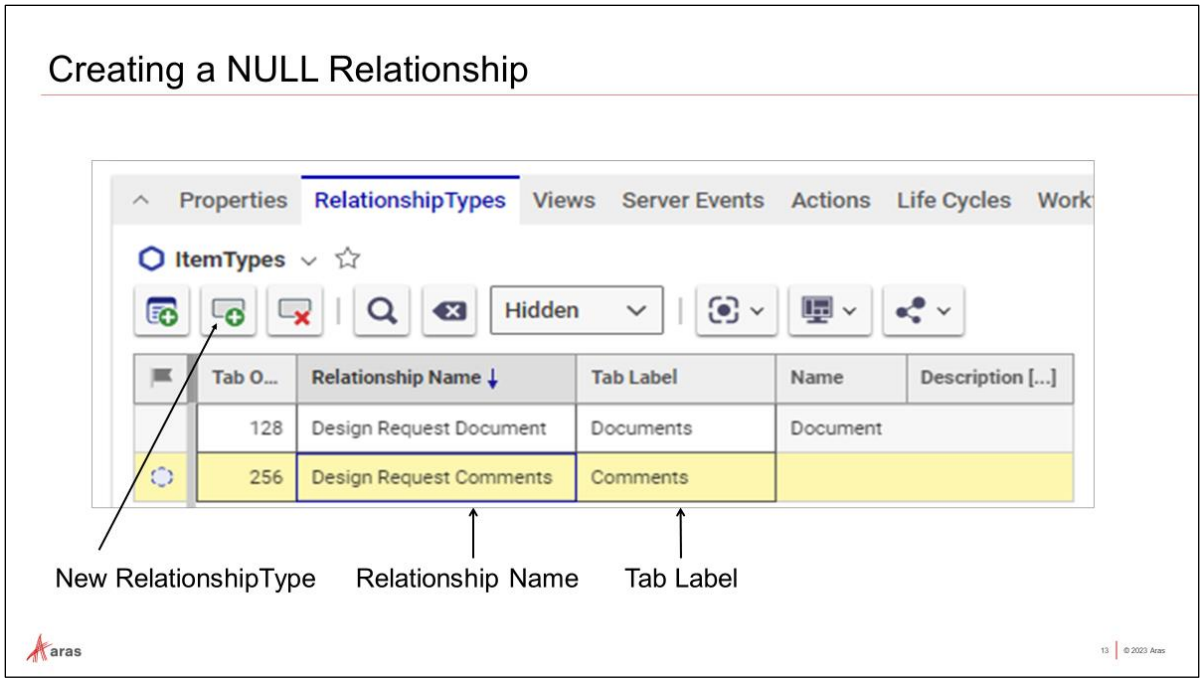
1. Navigate to Product Management > Development > Design Requests and create a new Design Request.
2. Look for the relationship tab labeled Documents.
3. Click the New Document button to add a relationship to a new Document.
4. In the new Document tab, enter a Document Number, Name, and select a Type.
5. Click Done.
6. For the Design Request, enter the **Request Owner**, and click Done.
7. On the new Document, click the Navigate button, and select Where Used to display all Items with a relationship to the Document Item.



Defining a NULL Relationship

NULL relationships are useful when you want to allow a user to add more than one piece of the same information to an Item, but do not need or do not have another ItemType to make a connection. It is important to add properties to a NULL Relationship ItemType and make them visible.

In the example above, a NULL relationship is used to define the ability for a User to add multiple remarks to an Item. A remarks String property can be assigned to the Relationship ItemType. Each time a User defines a new remark, an additional Relationship Item is created. No additional related item is necessary.



Creating a NULL Relationship

Null relationships are created on the RelationshipTypes tab. A NULL Relationship does not have an associated ItemType so the Name column will be blank for this RelationshipType. In this example, we use the Design Request ItemType.

Try it … Create a NULL Relationship

We want to allow multiple comments to be entered on Design Requests; therefore, in this exercise, we create a NULL relationship with a String Property to hold them.

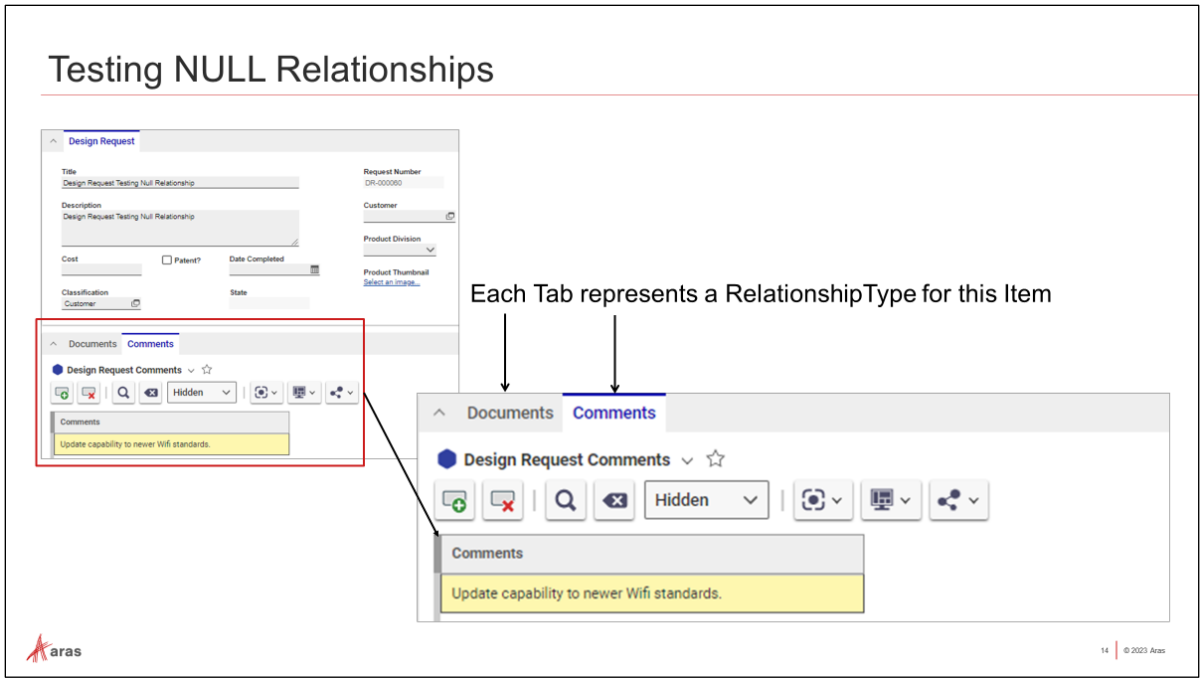
1. Return to the **Design Request** ItemType and open it for editing.
2. On the RelationshipTypes tab, click the New RelationshipType button to create a new relationship row.
3. Enter the following:

Tab Order	Relationship Name	Tab Label
256	Design Request Comments	Comments

4. Click Done.
5. Now, open the new **Design Request Comments** ItemType for editing.
6. Add a new Property, as follows:

Name	Label	Data Type	Length	Hidden2	Width
_comments	*Comments*	*String*	*255*	*unchecked*	*350*

7. Click Done.



Testing NULL Relationships

In our example, we added a NULL relationship to Design Requests. Test the NULL relationship by adding a new comment to a Design Request.

Try it … Test the NULL Relationship

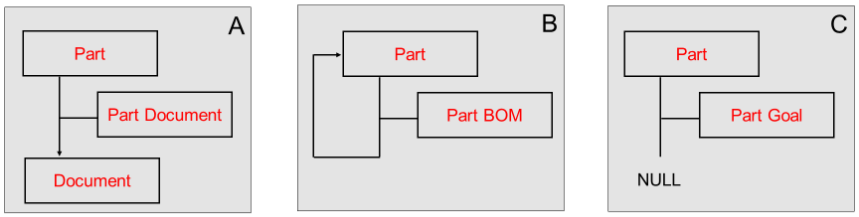
1. Navigate to Product Management > Development > Design Requests and create a new Design Request Item.
2. Enter at least a Request Owner since it is required to save.
3. Select the Relationship tab labeled **Comments**.
4. Click the New Design Request Comments button to create new rows.
5. Enter text in the provided field, e.g.:

Comments
Update capability to newer Wifi standards
Check for fax capability

6. Click Done to save the Design request and the new NULL relationship.

RelationshipType Patterns

- The diagrams below show typical relationship patterns
  - Diagram A shows a common type of relationship, where the source/parent item (Part) has a relationship (Part Document) to a separate related/child item (Document)
  - Diagram B show a circular relationship, where the source/parent item (Part) has a relationship (Part BOM) to a related/child item of the same type
  - Diagram C shows a "null" relationship, where there is no related item. This pattern is useful when there is a need to track items (Values, Comments, etc.) that pertain only to the source item and would not be reused



aras

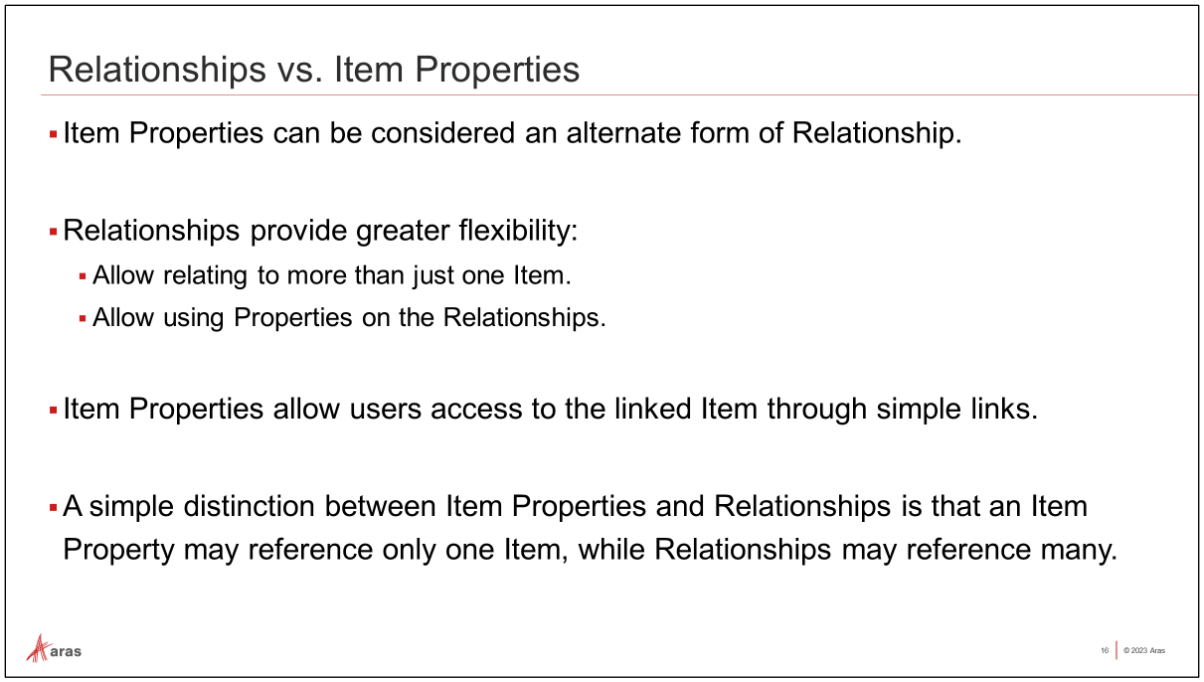
15 | © 2023 Aras

RelationshipType Patterns

Aras Innovator has three typical RelationshipType patterns, but there are others like relationships relating to other relationships.

- A Direct RelationshipType is a commonly used pattern and is best described as having a Source ItemType and a Related ItemType. See Diagram A above.
- A Circular RelationshipType simply references itself, such as a standard Aras Innovator Part ItemType. Viewing the Part BOM RelationshipType reveals the Source and Related ItemTypes are both a Part ItemType. In a Bill of Materials (BOM) hierarchy, Parts are connected to other Parts through the Part BOM RelationshipType.
- Null relationships must have at least one visible property in the Relationship ItemType or they will fail since they have no related ItemType. The Part ItemType contains a good example of a Null RelationshipType named Part Goal. The Part Goal RelationshipType does not have a Related Item since we only need it to display properties in the relationship grid.

Relationships may also be linked to other Relationships. An example is the (Part) BOM Substitute in the standard PE Solution. The Part BOM Relationship contains a link to the BOM Substitute Relationship which contains a pointer to a Related Item Part used as a substitute in form, fit, and function for that BOM entry. To see this Relationship example, view a Part BOM entry (using a right mouse-click) on a Part and then select the Substitutes tab.



Relationships vs. Item Properties

Earlier in the course, we discussed using properties of data type Item to link one item to another. An Item property provides reference to a single Item (and its foreign properties).

Relationships can provide more flexibility and more configuration options, e.g., one-to-many relationships to other Items using the Relationship Grid.

Some factors to consider when using a Relationship vs. an Item property:

- If a Foreign Property can be used to display related data, it is more efficient than using a relationship (fewer database tables required).
- If we want to offer users the convenience of selecting an item displayed as a hyperlink, we use an Item Property.
- Using multiple Item properties on a single parent Item (e.g., Manufacturing Plant 1, Manufacturing Plant 2, and Manufacturing Plant 3) makes the data model less flexible.
- Relationships give greater flexibility, allowing access to more than just one property value.
- Relationships provide one-to-many connections, Item properties only provide one-to-one.
- Generally, use an Item property when only one Item is referenced and use a relationship for referencing more than one item.

Summary

In this unit, you learned how to relate one ItemType to another ItemType using a Relationship. You also learned how to use NULL relationships to solve certain problems.

- You should now be able to:
- Define Relationships between ItemTypes
- Understand the Relationship ItemType
- Understand the RelationshipType
- Describe Typical Relationship Patterns
- Describe the Benefits of NULL Relationships

Review Questions

1. What is the difference between a RelationshipType and a Relationship ItemType?

2. Which one allows me to create new custom properties?

3. What is the advantage of using a Null relationship?

Lab Exercise

Goal

Create a relationship between two or more Items, assign the appropriate configuration, and define and use a Null Relationship.

Scenario

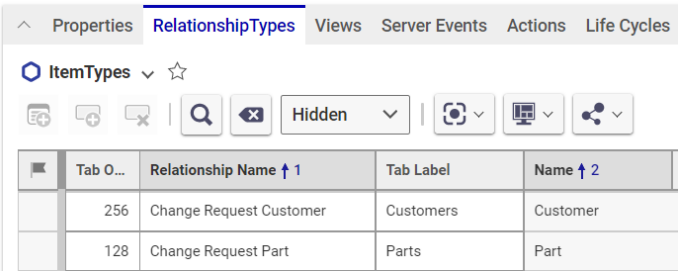
The original requirements for the Change Request system stated:

- Establish a required connection to a Customer.
- Allow a user to attach Parts to a request.
- Log one or more review notes made by the Change Review Board for tracking progress.

In this exercise, you establish relationships from the Change Request to a Customer Item as well as Part Items. You also add the ability to create review notes for a Change Request using a Null relationship.

Steps

1. Open the **Change Request** ItemType for editing and set the Default Structure View field to **Tabs On**.
2. Open the RelationshipTypes tab and using the Add ItemTypes button, add two new RelationshipTypes (Part and Customer).
3. For each Relationship, enter a Relationship Name and Tab Label. See example below.



4. Click Done.

5. A Change Request requires a relationship to a Customer; therefore, open the **Change Request Customer** RelationshipType and modify it so that Min Occurs is set to a value of *1*.
6. Parts related to a Change Request should only be created or selected.
7. Next, create a new Change Request Item. Verify the Parts and Customers tabs are visible in the Relationships grid.
8. Add at least one Customer (e.g., Delphi Corporation).
9. Now add a Part Item.
10. Create some additional Change Requests with Customer and Part relationships to test your work.

Capture Review Notes

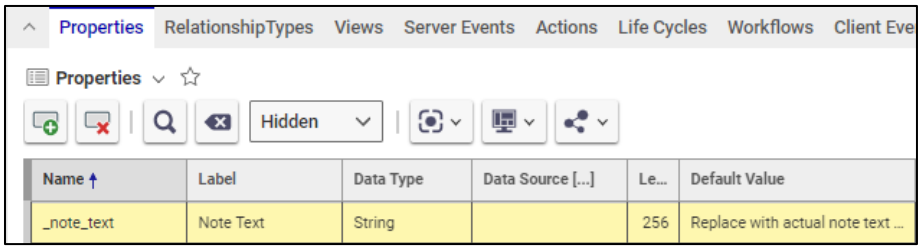
Next, we want to add the ability to capture Review Notes in the Change Request.

You accomplish this using a Null relationship.

1. Open the **Change Request** ItemType for editing.
2. On the RelationshipTypes tab, click the New RelationshipType button to create a new RelationshipType not pointing to any other ItemType. Set the Relationship Name to *Change Request Memo*, the Tab Label to *Review Notes*. The Name field will remain empty, as shown below.

Tab O...	Relationship Name 1	Tab Label	Name 2
256	Change Request Customer	Customers	Customer
384	Change Request Memo	Review Notes	
128	Change Request Part	Parts	Part

3. Save.
4. Open for editing the ItemType named **Change Request Memo** that was automatically generated by defining the new RelationshipType.
5. Add a String property to hold the Review Note text.



6. Set the width to *350* and click Done.
7. Create a new Change Request Item and note the new tab presented in the user interface.
8. Use the Review Notes tab to create a new Review Note, enter text, and click Done.

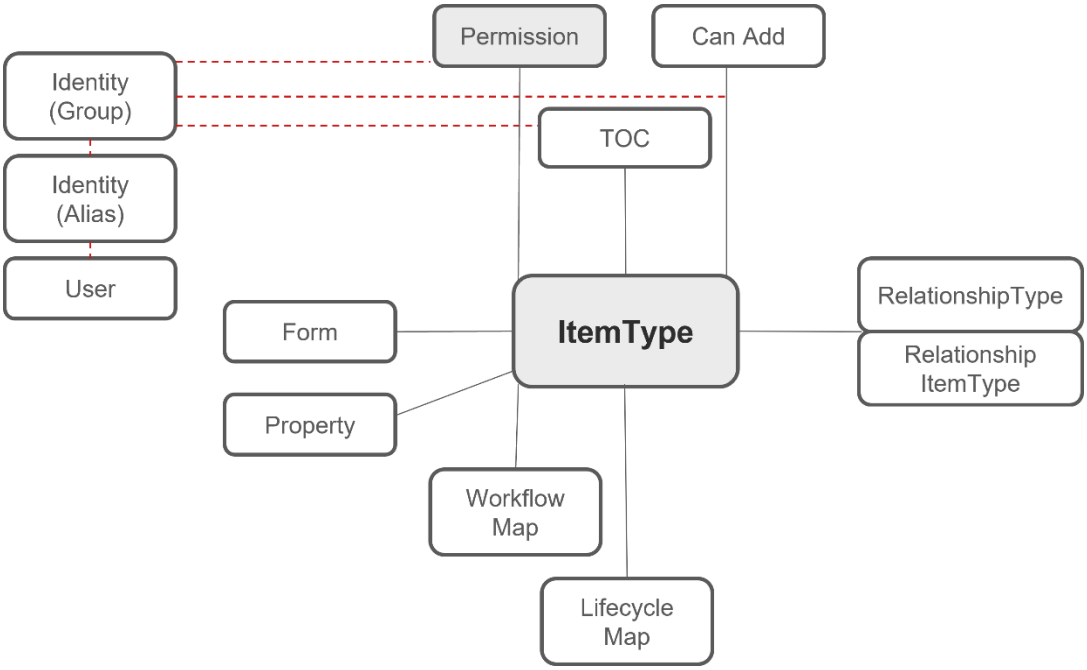
Unit 9 Assigning Security

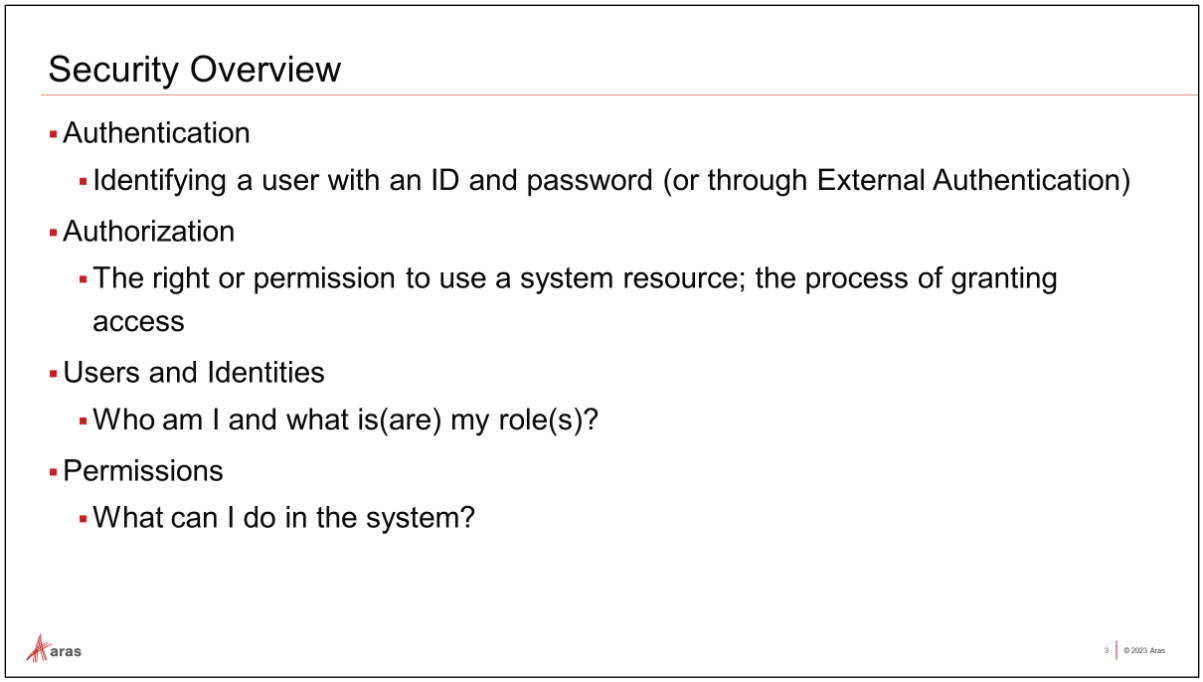
Overview

In this unit, you define security so that only authorized users have access to a solution by configuring the system to grant specific permissions to groups of individuals.

Objectives

- Explore the Aras Innovator Security Model
- Create Permissions
- Configure Special Permissions
- Create Private Permissions
- Assign Team Permissions
- Understand Password Policies





Security Overview

Authentication

Authentication provides a way of identifying a user, typically by having the user enter a valid username and password before access is granted. The process of authentication is based on each user having a unique set of criteria for gaining access. The server compares a user's authentication credentials with other user credentials stored in a database. If the credentials match, the user is granted access to Aras Innovator. If the credentials are at variance, authentication fails, and access is denied.

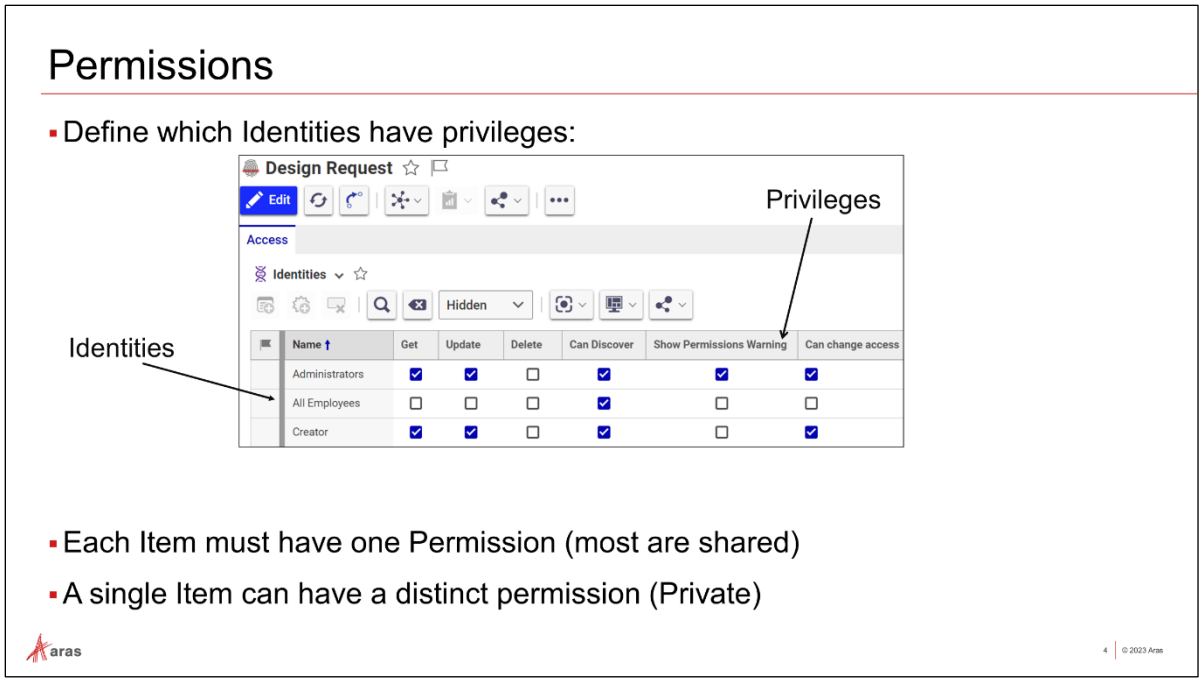
Authorization

After logging into a system, for instance, the user may try to create or edit Items. The authorization process determines whether the user has the authority to perform such actions. Authorization is the process of enforcing policies: determining what types or qualities of activities, resources, or services a user is permitted. Usually, authorization occurs within the context of authentication. Once a user is authenticated, the user may be authorized for different types of access or activity.

User Identities and Permissions

Aras Innovator provides security Authentication with configured User records as well as Authorization through Permissions.

In this unit, you learn how to create and configure Permissions.



Permissions

A Permission defines what access a user or group has to an Item. A Permission identifies one or more Identities and the rights and privileges assigned to each Identity, such as the ability to view, edit, etc.

Get Access

Enables Identities to search and open Items.

Update Access

Enables Identities to edit existing Items.

Delete Access

Enables Identities to delete Items.

Can Discover

Allows the user to search for items and view limited information about them in the search grid, but not view the detailed information that would be present on the Item form.

Show Permission Warnings

Permission Warnings inform users about not having access to Items.

Can change access

Enables Identities to select or create different permissions for an Item.

Permissions are assigned to an ItemType to define the default security for all Items of that type. Permissions can also be assigned to LifeCycle states to change the security of an Item over its lifetime.

A system property named permission_id identifies the Permission for the current Item being viewed or edited by a user. An Item can only have one Permission active at any one time, but many Items can share the same Permission definition.

Assigning Lifecycle Maps to subclasses of ItemTypes allows you to have different Permissions for different subclasses.

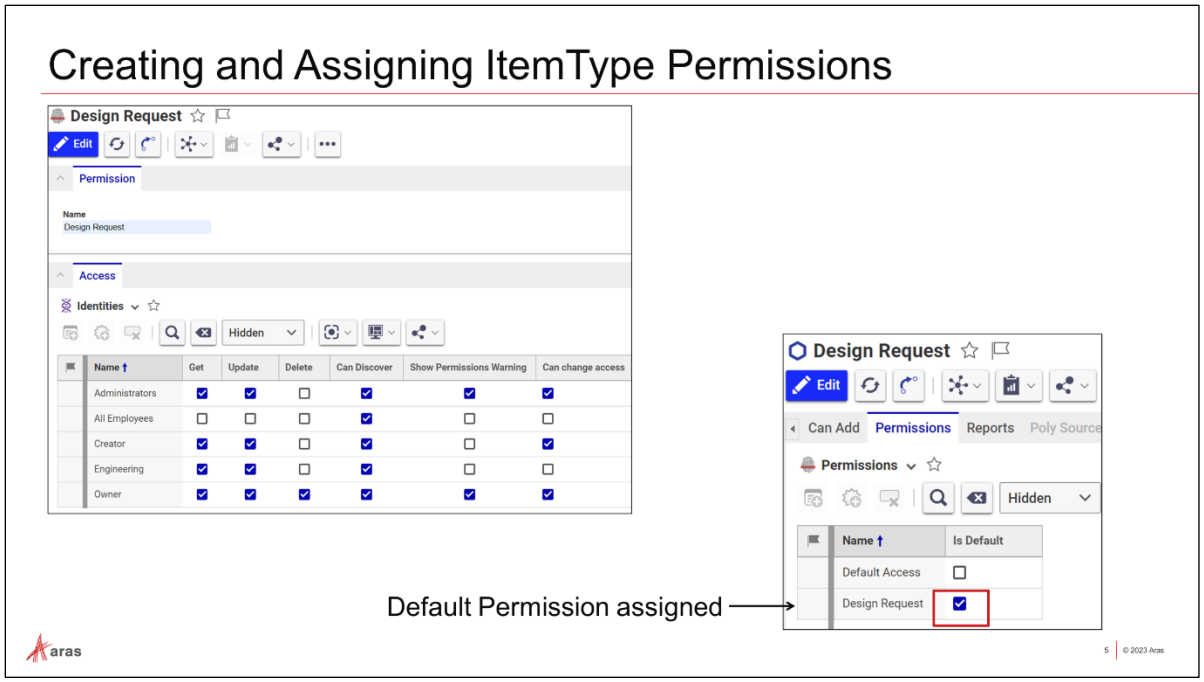
A special type of Permission, known as Private, allows you to define a Permission that is specific to one Item only and is never shared.

Note

Remember that Can Add and TOC Access are configured separately from a Permission. Can Add is set directly on the ItemType and TOC Access is configured in the TOC Editor.

Note

All identities (administrators and other identities) follow the permissions, with one exception. Only the super user 'root' identity overrides the permissions. The super user ‘root’ identity is used in specific cases, e.g. in case no one in the company can do a certain action anymore.



Creating and Assigning ItemType Permissions

Permissions are assigned to one or more ItemTypes, on the Permissions relationship tab of ItemTypes. An ItemType must have one default permission set to allow the creation of new items. Additional non-default permissions can be assigned.

Try it … Create a New Permission

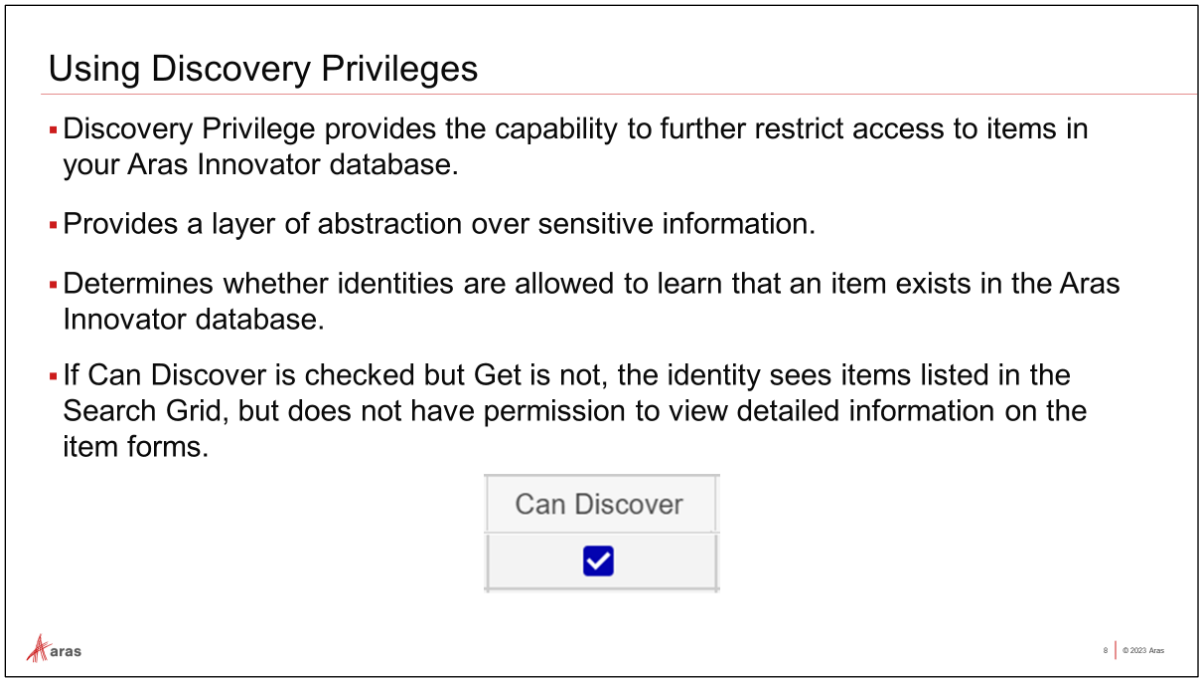
1. Navigate to Administration > Permissions and create a new Permission named *Design Request*.
9. On the Access relationship tab, grant access to the following Identities using the values as shown:

Name	Get	Update	Delete	Can Discover	Show Permissions Warning	Can change access
Administrators	☑	☑	☐	☑	☑	☑
All Employees	☐	☐	☐	☑	☐	☐
Creator	☑	☑	☐	☑	☐	☑
Engineering	☑	☑	☐	☑	☐	☐
Owner	☑	☑	☑	☑	☑	☑

2. Click Done.

Try it … Assign a new default Permission

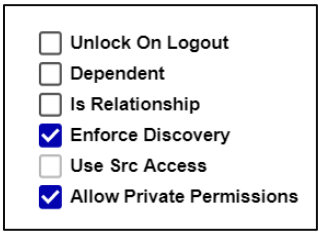
1. Open the **Design Request** ItemType for editing.
2. On the Permissions relationship tab, click the Add Permissions button.
3. Search for and select the **Design Request** Permission.
4. Click to enable the Is Default checkbox.
5. Deselect the Is Default checkbox for any other Permissions set.
6. Click the Done button.



Using Discovery Privileges

The primary function of the “Can Discover” privilege is to allow users to see items in the search grid but not open the items’ form. This restricts the visibility of the Items to only Properties that are not hidden. This feature provides a level of access control over sensitive information so that specific Identities can learn that an item exists.

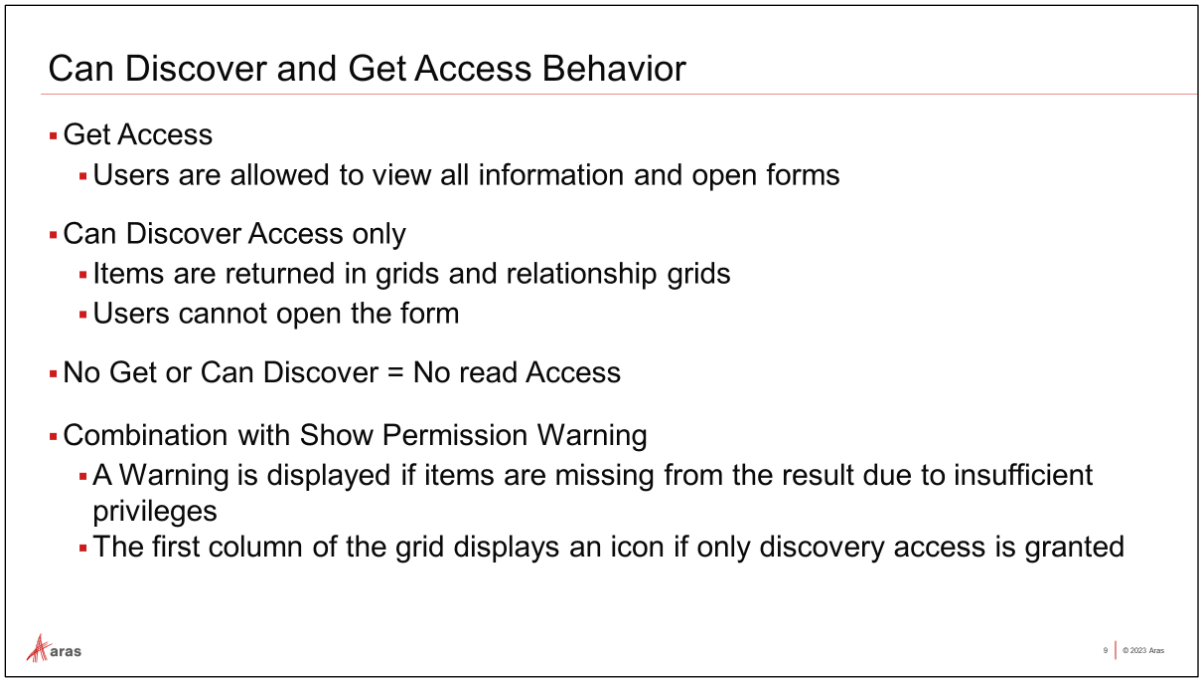
To enforce the Discovery privilege to discovery only, without Get privilege, you must also set the "Enforce Discovery" option in the target ItemType. When you uncheck Enforce Discovery on the ItemType, the Discovery privilege includes the Get privilege.



Display Rules in Relationship Grid

Relationship grids are impacted in a similar manner to the main search grid:

- Items without “Can Discover” access are shown with words “Restricted” in each column of the relationship grid. The User is unable to open the item or see any information about it. Items are displayed in this manner, so all users see a complete number of items in the grid. This prevents confusion and eliminates any chance restricted information is misunderstood as incomplete information.
- Items with “Can Discover” but without “Get” access have the Limited Permissions Icon in the first column, like in the main search grid.



Can Discover and Get Access Behavior

Both Can Discover and Get Access privileges allow you to define the amount of information that is viewed about an Item. It is important to understand how these two privileges work together.

If Get Access is set to true, users can view all information and open forms for the item. Identities with Get Access automatically receive Can Discover Access.

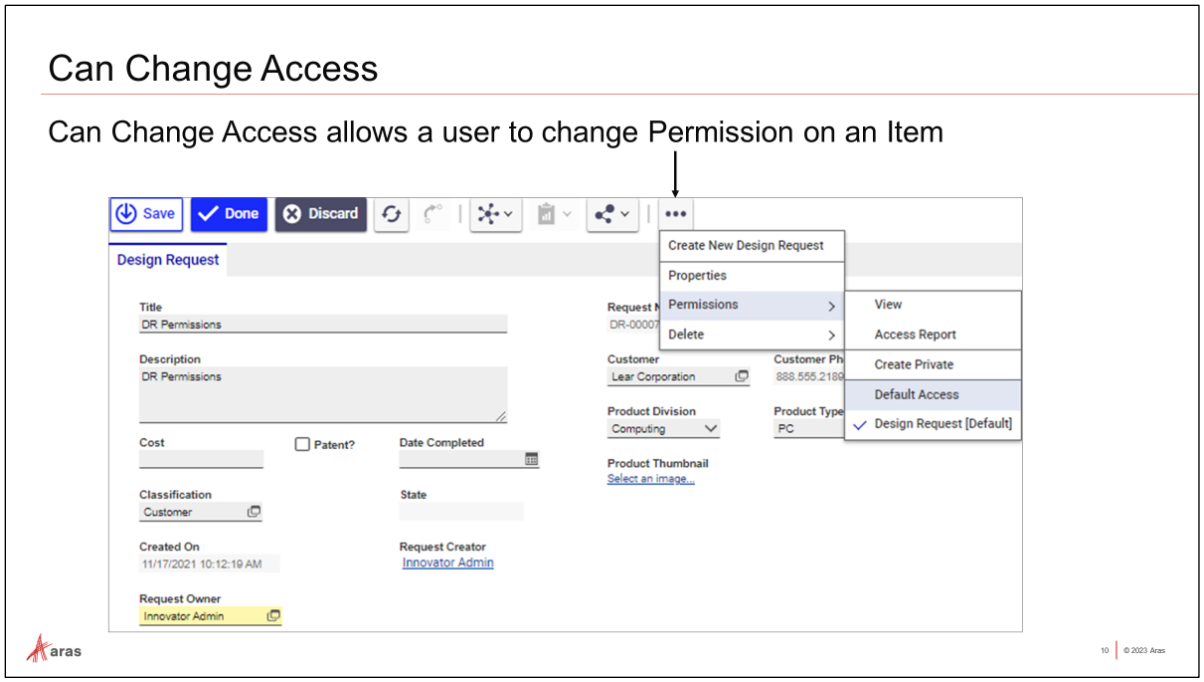
If Can Discover Access is set to true, but not Get Access:

- Items are returned in the main Search grid and the Relationship grid when they match the search criteria
- Users are restricted to viewing only the information returned in the grid and will not have the ability to open the form.

If the Show Permissions Warning is set to true and there are missing items from a search query:

- A warning icon displays.
- Warning text displays indicating a permission restriction.

If neither Get nor Can Discover Access is granted, Items are not returned in the main grid or the relationship grid even if they match the search criteria. A Warning displays indicating a permission restriction if Show Permissions Warning is set, and there are items missing from the result due to insufficient privileges.



Can Change Access

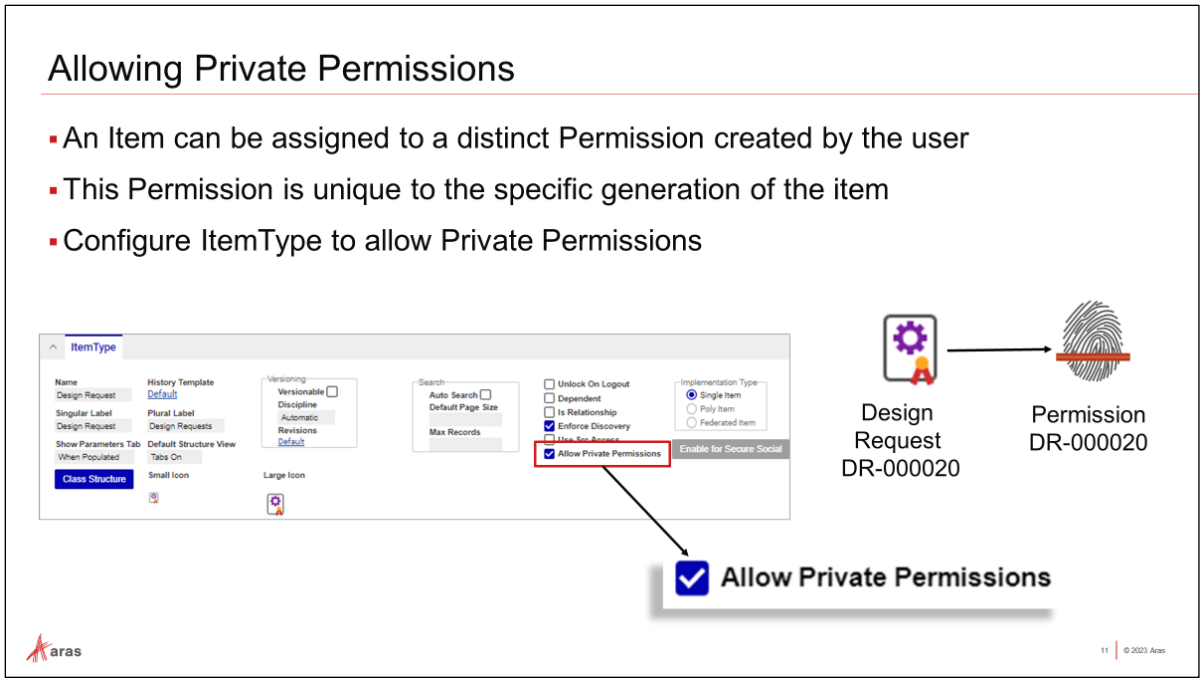
If the Can Change Access permission is set to true for an Identity, users of that Identity, in Edit mode, can change the Permission of the Item.

The list of all existing, available Permissions is defined in the corresponding ItemType. These Permissions display as choices under the …More > Permissions menu.

Try it … Change Access to an Item

Use the following exercise to change the Permissions of a Design Request. Note that only Design Requests created after the change will have the new permissions. First, we'll see who has Can Change Access permissions for Design Requests.

1. Navigate to Administration > ItemTypes and open the **Design Request** ItemType for viewing.
2. On the Permissions relationship tab, notice that there are two Permissions, Default Access, and Design Request, with the Design Request set to **Is Default**.
3. Open the **Design Request** permission and verify, that the Creator, Owner, and Administrators have permissions to change access.
4. Create a new **Design Request** item, select an **Owner,** and click the Save button.
5. Click the More menu button on the toolbar and select the Permissions sub-menu to display a list of available Permissions.
6. Select a different Permission from the sub-menu, **Default Access**.
7. Click Done.



Allowing Private Permissions

A Private Permission allows a user to change the access rights for the current Item. The configured Permission is unique to this single Item.

For example, an Identity (such as an auditor) may need access rights for a limited time. A private permission can be set to allow the auditor temporary access to items.

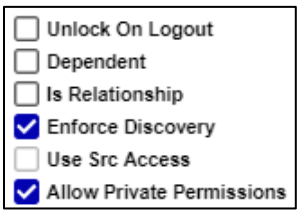
The Private Permission imposes a new set of access rights to this item, which is different than the default, and other permission sets, on an ItemType. The user creating the Private Permission must belong to an Identity with Can Change Access set on the default permission to invoke this option.

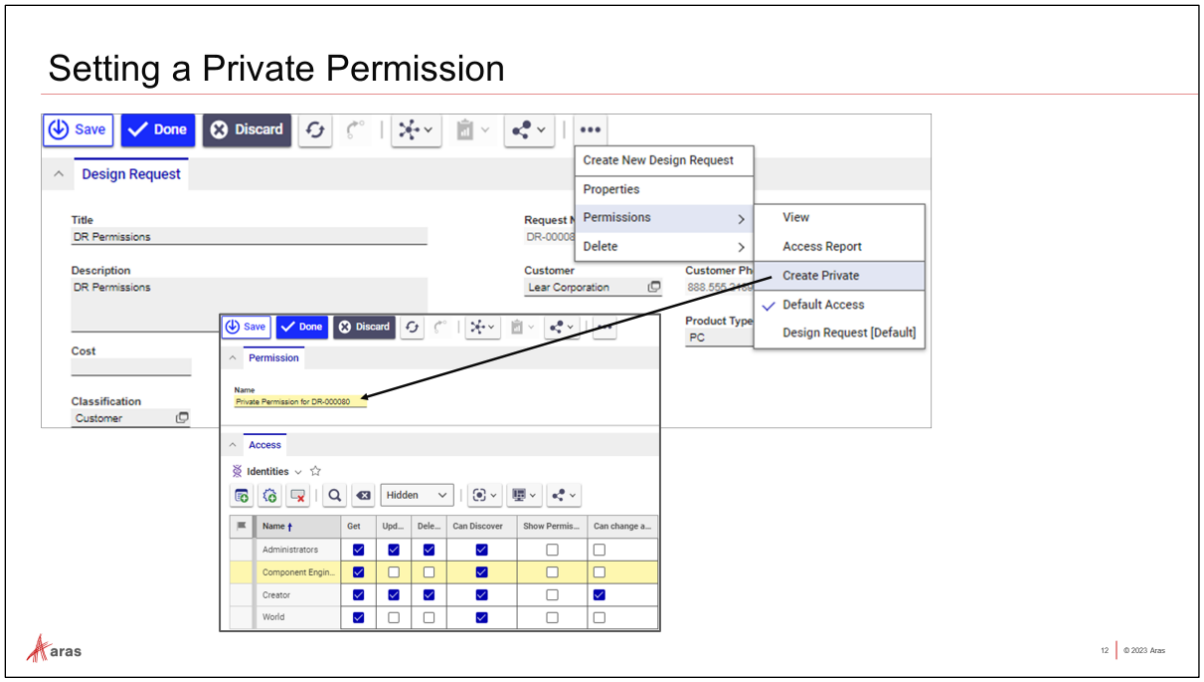
If Permission of the item is changed later to the default or other Permission assigned to the ItemType, the Private Permission is deleted.

To enable the use of private permissions for any ItemType, two factors must coexist. Allow Private Permissions check box must be checked on the ItemType form and an identity must be granted Can Change Access on the default permission.

Try it … Configure an ItemType to Allow Private Permissions

1. Navigate to Administration > ItemTypes and open the **Design Request** ItemType for viewing.
2. In the ItemType form, ensure Allow Private Permissions is enabled.





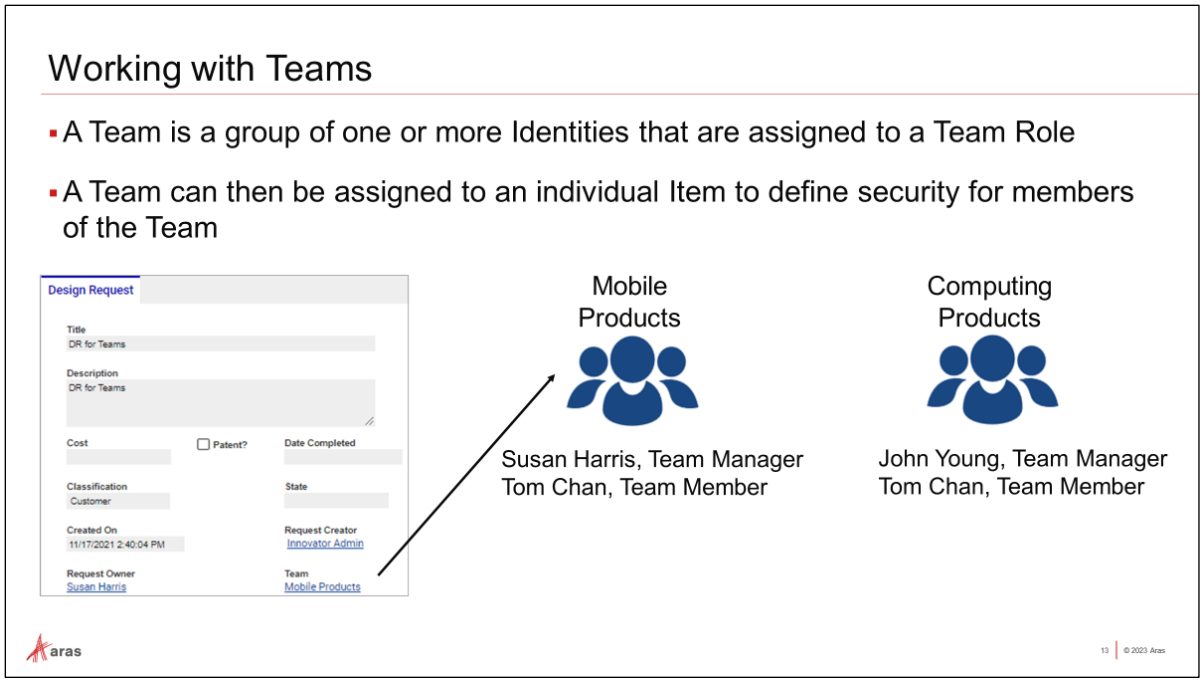
Setting a Private Permission

To set a Private Permission, the Identity must first have the Can Change Access permission. Combined with Allow Private Permissions enabled for the ItemType, you can choose between all related permissions for the ItemType. This also allows you to create a new, private Permission set for one instance of the ItemType.

Try it … Set a Private Permission

1. Create a new **Design Request** adding *Administrators* as the **Owner** and save it.
2. Click the More menu button, then click Permissions > Create Private to create a new Private Permission for this Design Request.
3. Change the access rights by adding the *Component Engineering* Identity and give it both Get and Can Discover Access.
4. Give the new Private Permission a name. Name it: *Private Permission for DR-0000XX*.

   Where **XX** = your DR Number.
5. Click the Done button to save the new Permissions Item.
6. Return to the Design Request Item tab and click Done to save the Private Permissions with the DR.



Working with Teams

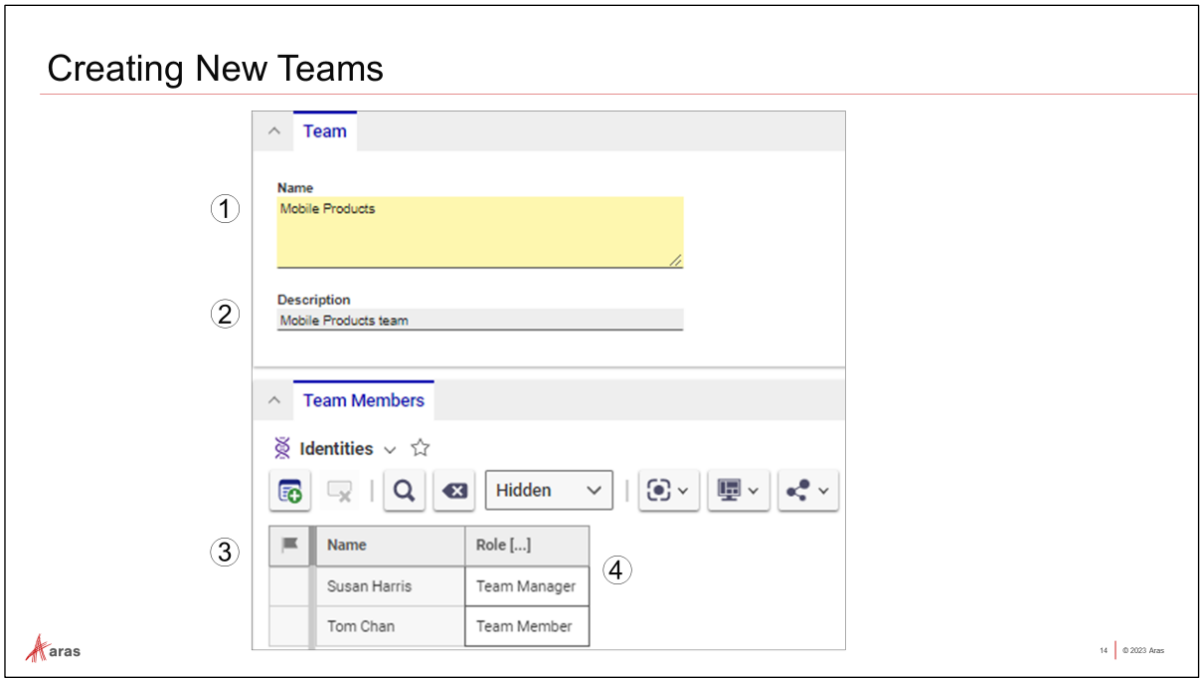
A Team is defined as a group of one or more Identities that are assigned as members. Each Identity that is included in a Team can be assigned a Team Role that indicates the member's responsibility for an Item. A Team Role can then be assigned in a Permission to allow for dynamic security based on Team membership. Teams and Team Roles can also be referenced in LifeCycle Maps, Workflow Maps, and in most other situations where Identities can be used. Aras provides a standard set of Team Roles, but you can also define your own.

In the example above, two Teams have been created and one is assigned to a Design Request. The Mobile Products Team includes two members, assigned to the roles of Team Manager (Susan Harris) and Team Member (Tom Chan). Another Team named Computing Products has been defined with different Identities for the same Team Roles.

If a Permission is created that uses the Team Manager role – the Identity of that role is determined by which Team was selected for the Item. Susan Harris is defined as Team Manager if the Mobile Products team is selected for an item. John Young is the Team Manager if the Computing Products team is selected instead.

Standard Team Roles include Team Manager, Team Member, and Team Guest. If a role is not assigned to a member of a Team, they assume the Role of "Team". If a group identity is assigned to a team role, all members of the group receive the same permissions.

The system property team_id must be displayed on a Form to allow a user at runtime to select the appropriate Team.



Creating New Teams

A Team Item provides the following fields and relationship properties:

①	**Name**	The Name of the Team.
②	**Description**	The Details about the Team.
③	**Team Members Name**	An Identity that is defined as a member of the Team. This can be a Group or Alias Identity.
④	**Role**	A Team Role. Can choose Team Manager, Team Member, or Team Guest. If you leave it blank, the Identity assumes the role of Team.

Try it … Create a New Team

This exercise is required for the lab in this unit.

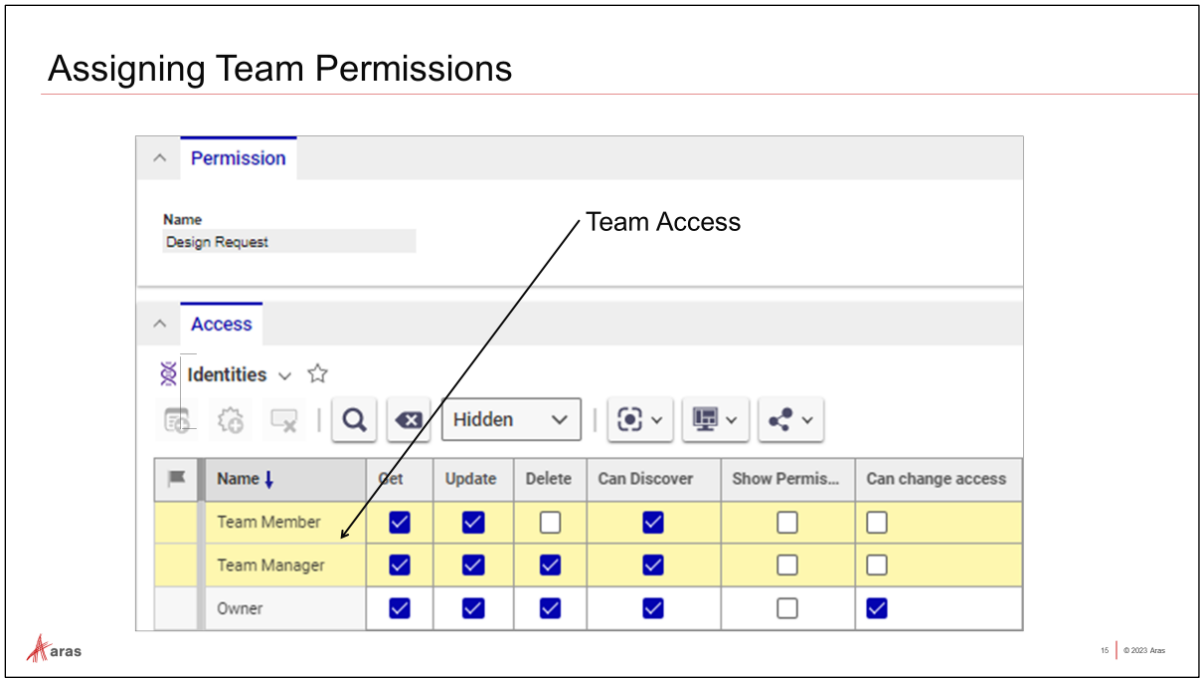
1. Navigate to Administration > Teams and create a new Team.
2. Enter *Mobile Products* for the Team Name and enter a Description.
3. On the Team Members relationship tab, assign Team Roles using the following values:

Identity Name	Team Role
Susan Harris	Team Manager
Peter Smith	Team Member

4. Click the Done button.
5. Create another Team named *Computing Products*, and define the following members:

Identity Name	Team Role
John Young	Team Manager
Peter Smith	Team Member

6. Click the Done button.



Team Permissions

By creating Teams, you can supplement the robust security system provided in Aras Innovator. Team Roles can be assigned to a Permission to enhance access to Items. The actual user that represents the Team Role gains access based on the selected Team for an item.

Once Permissions are edited to include Team Roles, and the Team property is added to an item, access for team members is enhanced.

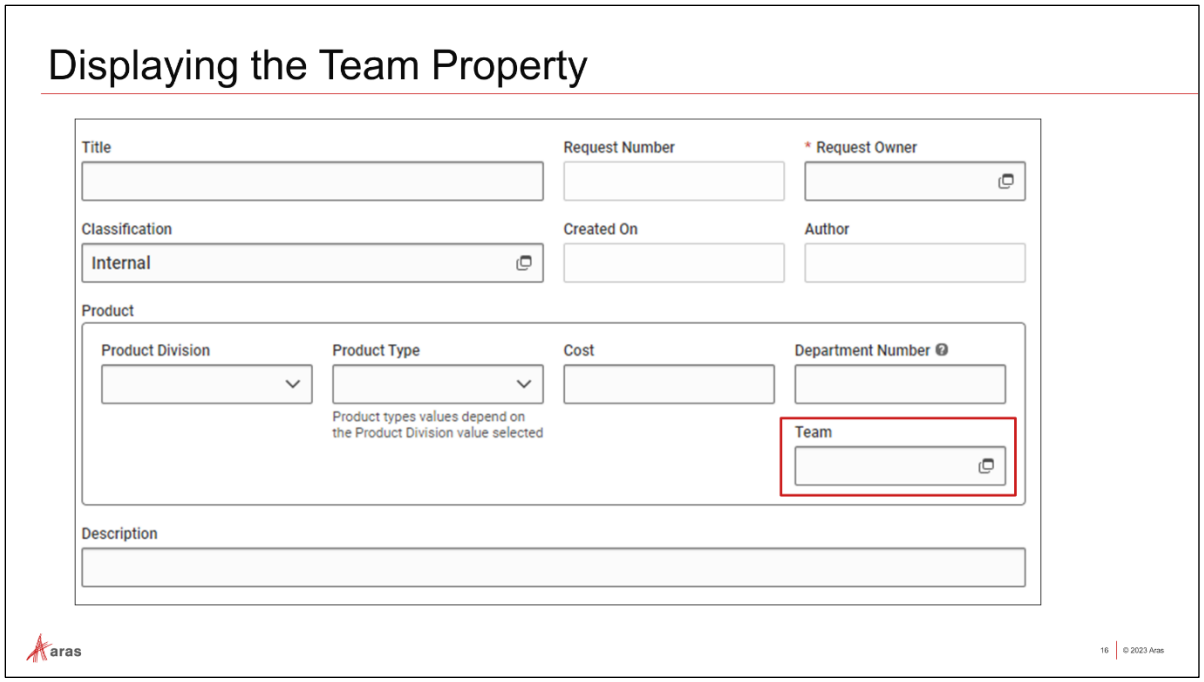
In this example, the Design Request Permission has been modified to include both the Team Manager and the Team Member roles with appropriate security rights based on their level of responsibility.

Try it … Assign Team Permissions

1. Navigate to Administration > Permissions and open the **Design Request** Permission for editing.
2. On the Access relationship tab, grant access to the following Identities using the values as shown.

Name	Get	Upd...	Dele...	Can Discover	Show Permis...	Can change a...
Team Member	☑	☑	☐	☑	☐	☐
Team Manager	☑	☑	☑	☑	☐	☐

3. Click Done.



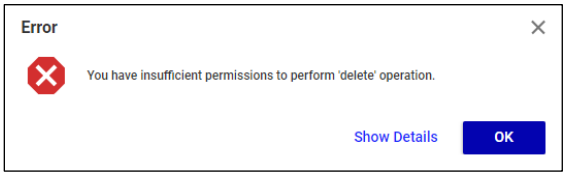
Displaying the Team Property

The Property team_id is an Item Property and it can be placed on a form like any other property. Once it is placed on a form, it can be used to select a Team for an Item.

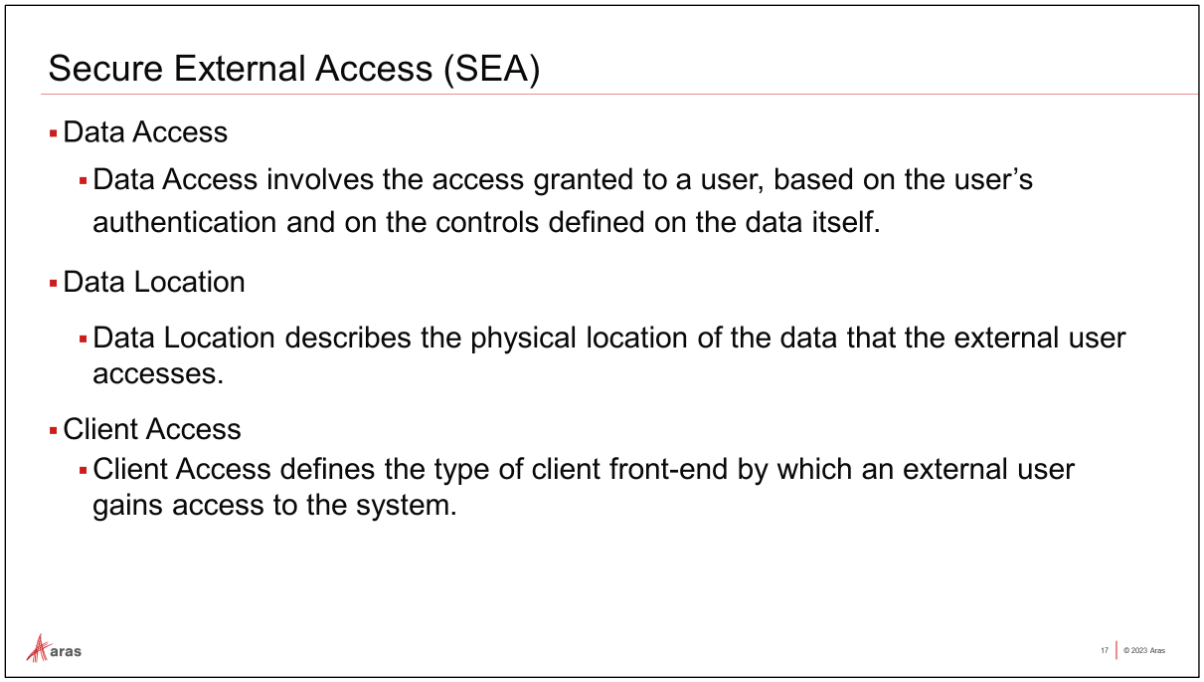
Try it … Display the Team Property on the Design Request Form and Test Team Permissions

In this example, the Design Request Form is modified to allow for the selection of a Team. In addition, we can check that Tom Chan, as a team member, doesn't get the permission to delete a Design Request.

1. Navigate to Administration > Forms and open the **Design Request Internal** Form for editing.
2. Drag and drop the *Team* property to the column containing *Department Number*.
3. Click Done.
4. To test: Create a Design Request item, enter *Diane Prescott* as the **Owner**.
5. Set Classification to *Internal* and choose the **Mobile Products** Team.
6. Click Done and take note of the Request Number.
7. Using a different browser, sign in as **Peter Smith** (psmith, innovator), a Team Member. Open a **Design Request** search grid and click Search.
8. Right-click on the Design Request you just created and select More > Delete.
9. In the Confirm message dialog window, click Delete.
10. In the resulting Error message window, click OK.



11. You can logout (as Peter Smith) and close the browser.



Secure External Access (SEA)

Secure External Access (SEA) with the Aras PLM Platform connects external partners to PLM data, while providing highly secure access controls for IP. Conceptually, SEA provides direct data access, with no artificial data packaging, allowing natural access to data and processes.

After appropriate access is granted by the main company's IT administration, external users can access items and files, update items and files, and participate in processes such as workflow and collaboration.

These three dimensions are Data Access, Data Location, and Client Access. Each of these represents a specific capability of the Aras PLM Platform.

Data Access

The Data Access dimension involves the access granted to a user, based on the user's authentication and on controls defined on the data itself. There are three types of access control supported by the Aras PLM Platform, which in order of increasing sophistication are: Standard permissions, Mandatory Access Control (MAC Policy), and Domain Access Control (DAC).

- Standard Permissions: The Aras standard permission model is a form of Role Based Access Control and sets permissions based on the user's defined identities.
- Mandatory Access Control (MAC Policy): MAC Policy is the next level up and it is a form of Attribute Based Access Control. This type of access control is commonly used by government organizations such as the military. The basic premise of MAC Policy is that the clearance level of the subject (the user) must meet or exceed the classification level of the item (object) being accessed.
- Domain Access Control (DAC Policy): The most sophisticated level of access control is a form of Relationship Based Access Control, whereby security can be inferred based on the relationships of items to other items, rather than being specified directly as in MAC Policy.

Data Location

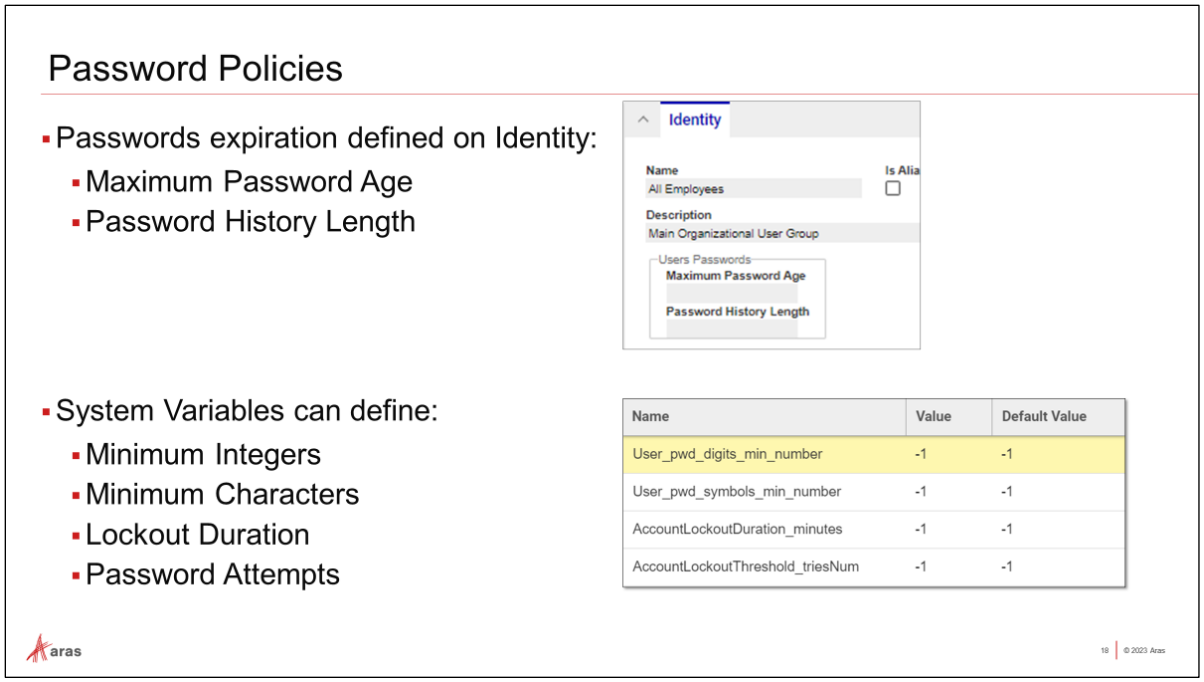
The Data Location dimension of SEA describes where the data that the external user accesses is physically located, either inside or outside your company's firewall. This configuration is implemented by your IT department and dependent upon your business use cases. The choice for your configuration tends to be strongly dictated by your corporate policy, which drives decisions about the other two dimensions.

- Inside the Firewall – Single Database: Data is contained in a single database inside of the firewall.
- Outside the Firewall - Multi-Database: This approach involves setting up a separate Aras server and database instance in a location outside the firewall. Either in an isolated network positioned between the Internet and the private network) or in the cloud.

Client Access

Client Access defines the type of client front-end that you use to gain access to the system.

The Standard Aras Web Client is the default type of client front-end: using the standard HTML client, suppliers and partners can make use of full functionality within Aras Innovator.



Password Policies

If you are not using External Authentication, then Password policies allow flexibility for administrators to configure and maintain User passwords.

Each Identity has passwords expiration settings that can be independently applied. When applied to a Group Identity (e.g. *All Employees*), those settings cascade down to all members of this Identity. You can either apply password settings in one top Group Identity and make all user Identities members of it (except any system users that might be used to allow things like the Aras Innovator Service to connect to Aras Innovator), or have several passwords expiration settings on different group Identities.

Maximum Password Age indicates that the password must be changed after this specified number of days. Password History Length specifies how many old passwords are saved and prevented from being used again when resetting a password.

Additionally, system variables are used to specify password restriction. You can view or change any of the following Variables.

User_pwd_symbols_min_number	defines a minimum number of characters for a password. -1 means there is no minimum number.
User_pwd_digits_min_number	defines the minimum number of integers in a password. -1 means there is no minimum number.
AccountLockoutThreshold_triesNum	defines the number of incorrect password tries before an account is locked out. Must be used with AccountLockoutDuration_minutes.
AccountLockoutDuration_minutes	defines the number of minutes before an account is released from being locked out. Must be used with AccountLockoutThreshold_triesNum.

Try it … View or Change Password Settings

3. Navigate to Administration > Identities and locate the alias Identity for Peter Smith.
12. Notice the available settings for Maximum Password Age and Password History Length.

Summary

In this unit, you learned about how to implement the Aras Innovator security model.

You should now be able to:

- Create Permissions
- Create Private Permissions
- Assign Discovery Privileges
- Assign Team Permissions
- Assign Password Policies and Permissions

Review Questions

1. Can the access to an item be controlled by a permission that is not listed in the Permissions tab of the ItemType template?

2. How would you allow someone to see an Item in the search grid but prevent them to open the item Form?

3. How can you block someone from the system after 3 failed logons?

Lab Exercise

Goal

Refine the security of the Change Request ItemType to allow only certain users to perform specific operations on new or existing Change Request Items.

Scenario

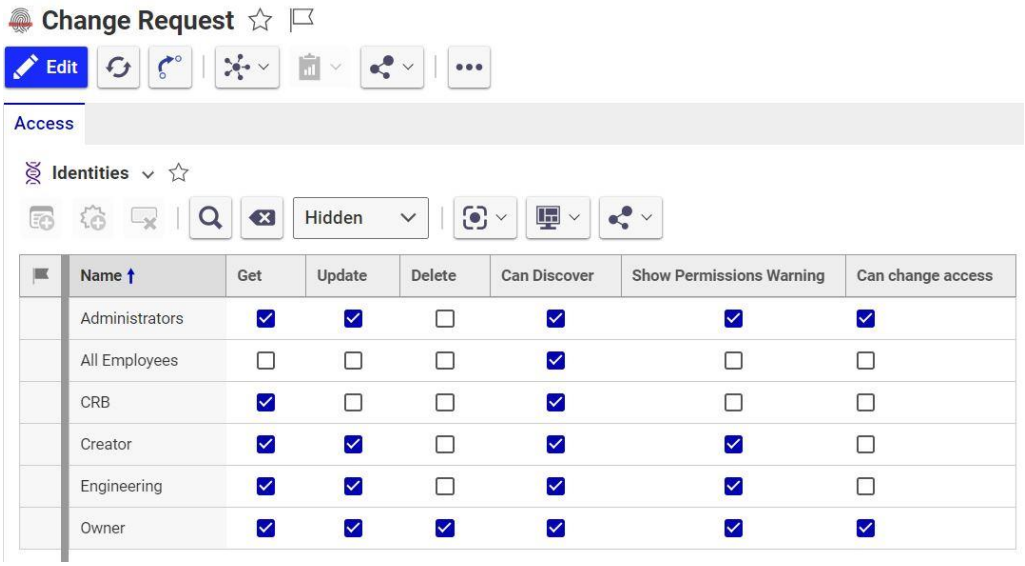
Refine the permissions for the Change Request ItemType.

Preparation

1. Before beginning this exercise, delete any existing Change Request Items you created in previous exercises. This will make it easier to see how making changes in security affects a user's ability to work in the system.

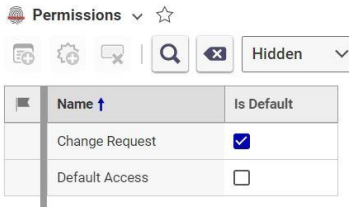
Permissions

1. Create a Permission named Change Request and configure the Permission as follows:



Name	Get	Update	Delete	Can Discover	Show Permissions Warning	Can change access
Administrators	☑	☑	☐	☑	☑	☑
All Employees	☐	☐	☐	☑	☐	☐
CRB	☑	☐	☐	☑	☐	☐
Creator	☑	☑	☐	☑	☑	☐
Engineering	☑	☑	☐	☑	☑	☐
Owner	☑	☑	☑	☑	☑	☑

2. Click Done.
3. Open the **Change Request** ItemType for editing, and select the *Permissions* tab.
4. Add the **Change Request** permission created above. Ensure you select it as the Default permission. Deselect the Is Default checkbox for any other Permission.



Name	Is Default
Change Request	☑
Default Access	☐

5. Currently the Can Add security setting for the Change Request ItemType is assigned to the **World** Identity. Replace the **World** Identity with the **Engineering** Identity. Ensure the "Can Add" checkbox is selected for the Identity:

Name	Can Add
Engineering	☑

6. Click Done and attempt to create a new Change Request as an Administrator.

7. Use a different browser and log into Aras Innovator as Peter Smith (*psmith*).
8. Create a new Change Request with Peter Smith as the Owner, add a related Customer and click Done.
9. Logout as Peter Smith.

To make it easier for testing, we want to allow Administrators to also be able to add new Change Requests to the system. You will need to log out and back in to see the permission added in the next step.

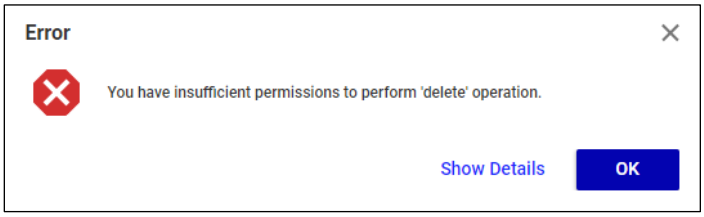
4. Edit the Change Request ItemType to allow Administrators to create Change Requests.

Display the Team Property on the Change Request Form and Test Team Permissions

1. Open the **Change Request** form for editing, add the **Team** property, move it to a desired location, and click Done.
2. Navigate to Administration > Permissions, open the **Change Request** Permission for editing, and on the Access relationship tab, grant access to the following Identities using the values as shown:

Name ↓	Get	Update	Delete	Can Discover	Show Permissions ...	Can change access
Team Member	☑	☑	☐	☑	☐	☐
Team Manager	☑	☑	☑	☑	☐	☐

3. Click Done.
4. Create a Change Request item, add *John Young* as the **Owner**, add the *Computing Products* team, click Done, and note the Request Number.
5. Using a different browser, sign in as Peter Smith (*psmith, innovator*), a Team Member. Open a search grid, right-click on the Change Request you just created, and select More > Delete.
6. In the resulting Confirm message dialog window, click Delete.



7. Logout (as Peter Smith) and close the browser.

This page is intentionally blank

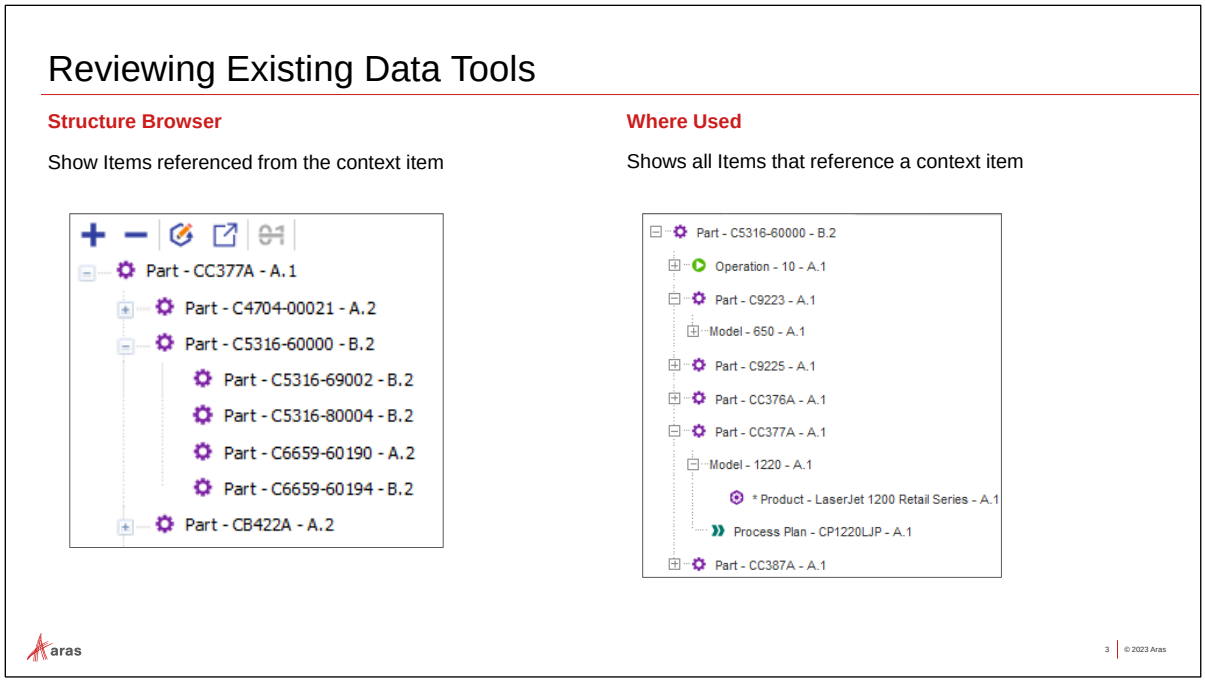
Unit 10 Visualizing Data

Overview

In this unit, you will learn how to present Item data to a user in an easily navigable format. Using a graphical interface, you will define and save complex reusable query definitions that retrieve property data from the Aras Innovator database. Query results are then mapped to a configurable Tree Grid View (TGV) that allows a user to visualize information in a hierarchical display. Knowledge of AML is not required to create query definitions, though a basic understanding of the data constructs is helpful.

Objectives

- Review Existing Data Tools
- Create a new Query Definition
- Create and Use a Tree Grid View
- Enable Query Parameters

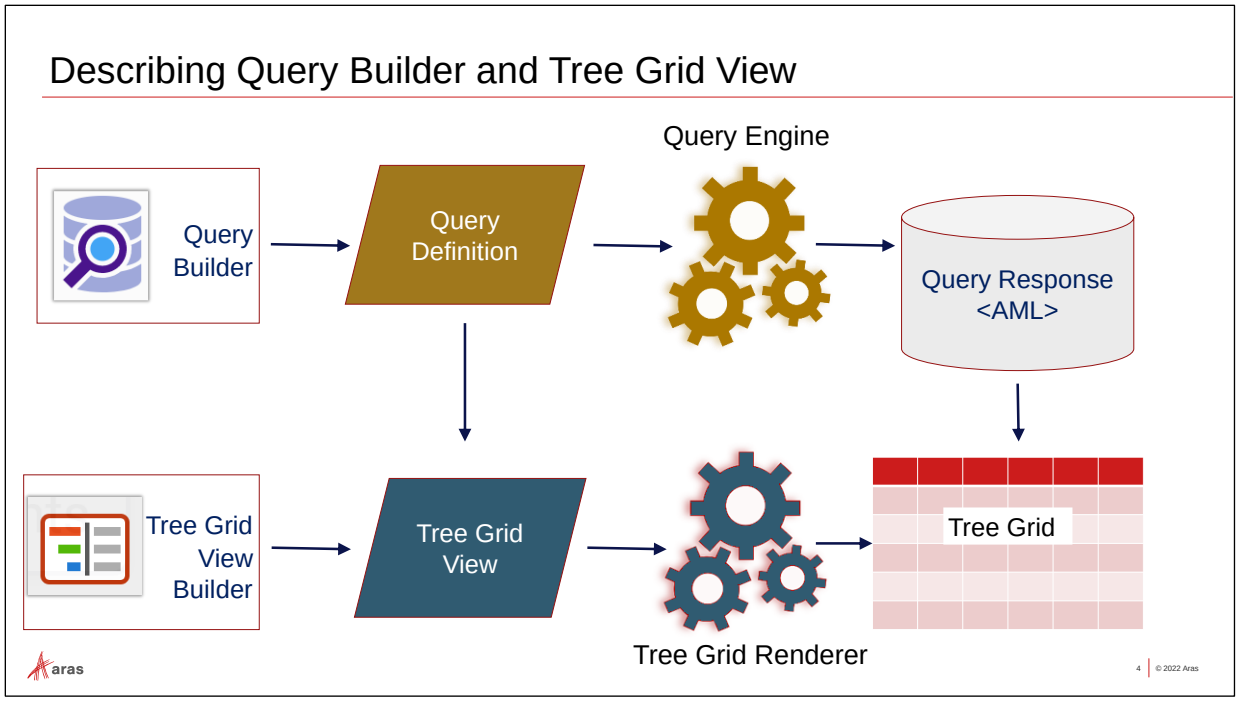


Reviewing Existing Data Tools

Searches (both Simple and Advanced) use an XML-based language, known as Adaptive Markup Language (AML). AML is converted to a Structured Query Language (SQL). An understanding of AML is not needed to work with the Query Builder.

Aras Innovator contains pre-built views that allow data to be displayed in an expandable tree-structure format. The Structure Browser, a top-down display and Where Used, a bottom-up display, shows all items that reference the current item in context.

This unit shows you how to create custom queries that retrieve item data and their relationship data. Users can then navigate from a source item to other related items or view items that are referencing the current item. Query Builder will allow the creation of queries that can be used with Tree Grid Views, Graph Navigation, Reporting and Access Control.



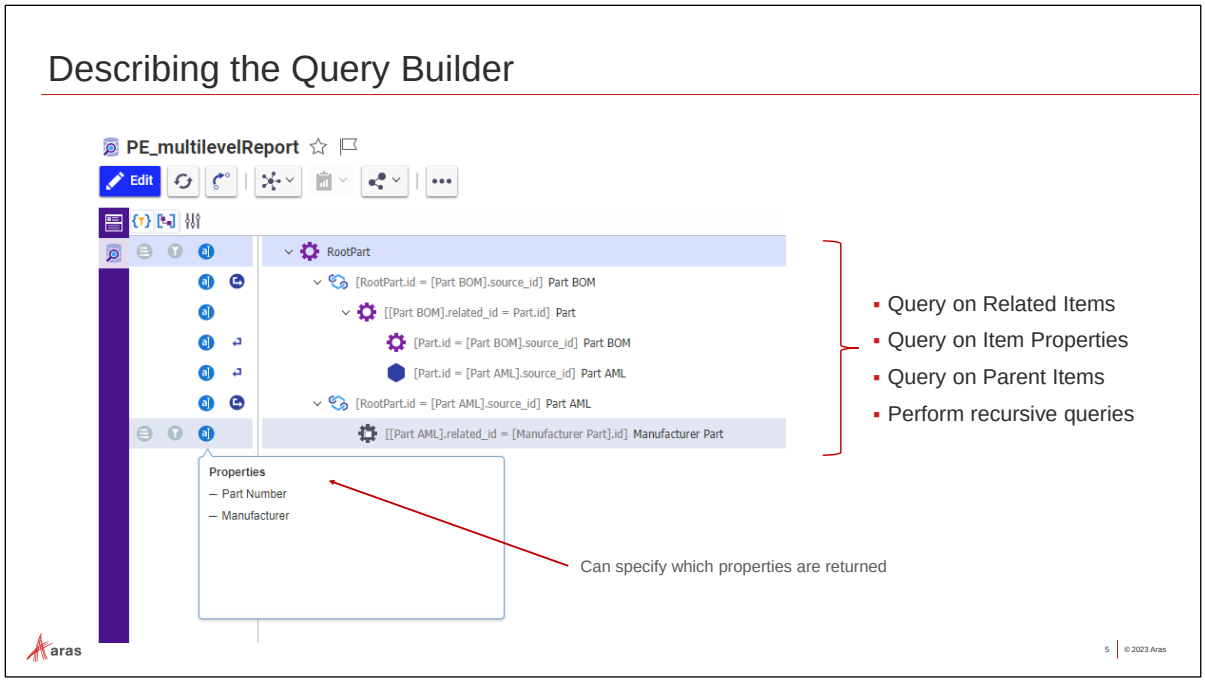
Describing Query Builder and Tree Grid View

The diagram above outlines the basic steps involved in displaying data in a Tree Grid View.

An administrator first defines a Query Definition using the graphical Query Builder editing tool that allows for the selection of items, relationships, and property data from the Aras Innovator database.

You create a separate Tree Grid View using the Tree Grid View builder to display the queried data in a series of hierarchical rows and columns. You use property data, retrieved from an associated Query Definition, to map to columns in the Tree Grid View. Please note that a Query Definition is separate from a Tree Grid View and the Query Definition can be reused or used for other items such as Graph Navigation, 3D Visualization, and Reporting.

The initial ItemType is the starting context item. When a user opens a starting context item, the Query Engine retrieves the item data, which is then rendered by the Tree Grid Renderer, and displays the resulting grid to the user.



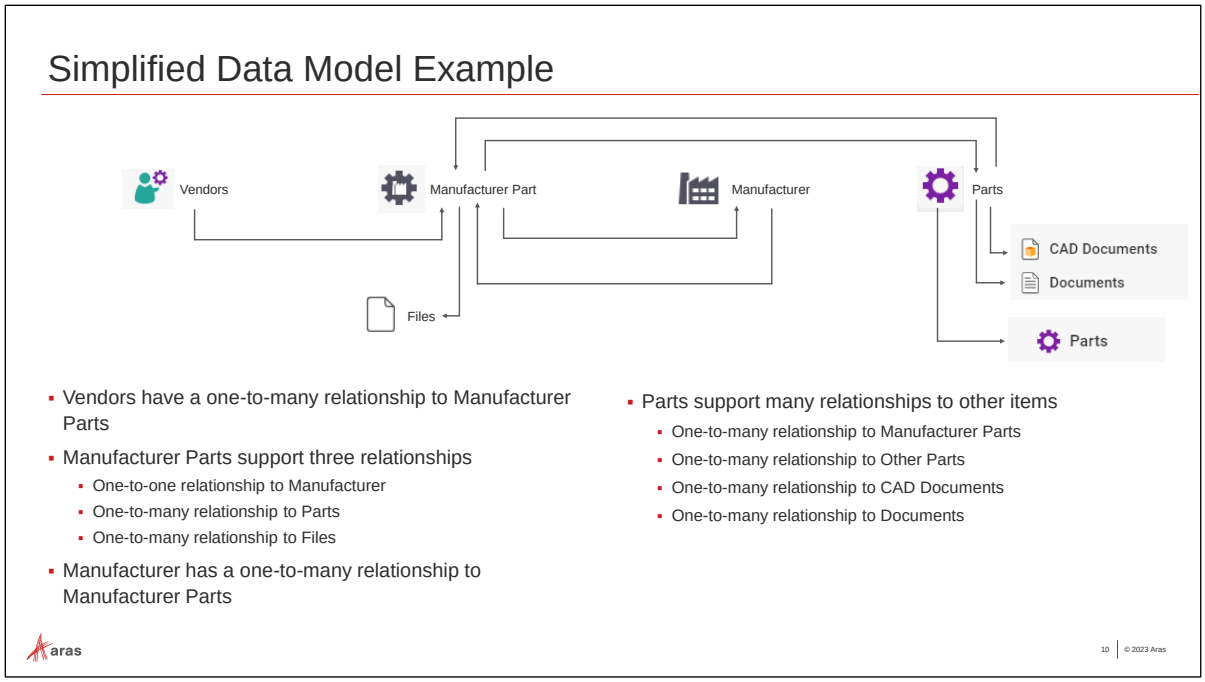
Describing the Query Builder Tool

Use the Query Builder to create reusable Query Definitions that you can then associate with one or more Tree Grid Views or use with Graph Navigation, 3D Visualization, and Reporting.

Query Builder uses a graphical interface which allows an administrator to choose which items, relationships, and properties are retrieved from the database.

With Query Builder, you can:

- Select the property data to be returned
- Build queries that return related Items based on a relationship
- Build queries that return item data based on Item datatype properties
- Perform where used queries that return referencing items
- Perform recursive relationship queries on nested data



Simplified Data Model Example

The out-of-the-box installation of Aras Innovator includes multiple categories to support part acquisition found under the Sourcing category of the TOC. There are a number of pre-defined relationships to support tracking which parts certain manufacturers make and which vendors they can be obtained from.

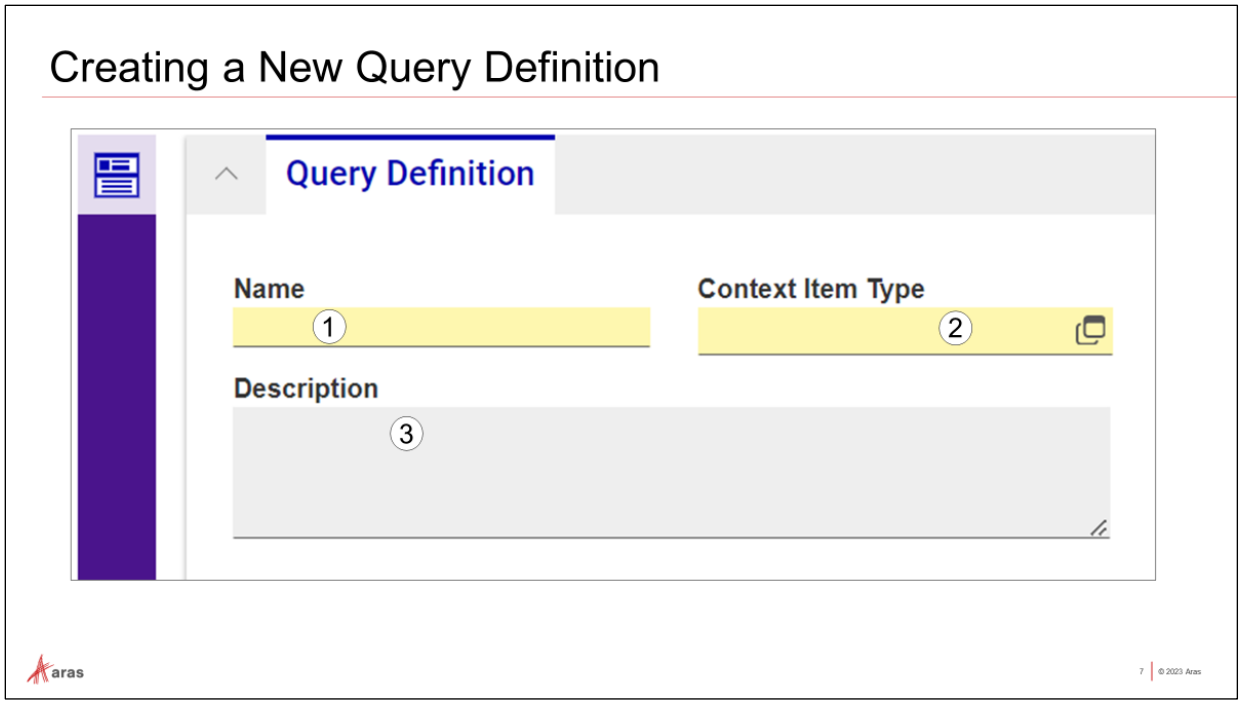
Each defined Part can have one or more Manufacturer Parts associated with it. For each Manufacturer Part, there is only one manufacturer who makes it, but there can be many vendors who sell it.

In previous units, you have created a one-to-one relationship on an ItemType by creating a Property, with the Data Type set to Item, pointing to a Data Source. Query Builder allows related items to be added to a Query definition and include properties of the selected item.

You also created one-to-many relationships on an ItemType pointing to another ItemType. You used the default relationship tabs to select and display any items from the other category of items. These related items, including their properties, can also be added to a Query definition.

The Try Its… in this unit will explore adding related items to a Query definition using all three ways:

- Using Item Property
- Using Relationship
- Using Referencing Item



Creating a New Query Definition

An administrator creates Query Definitions, found under the Configuration subcategory in the TOC.

The following fields display:

①	**Name**	The name of Query Definition.
②	**Context Item Type**	The root element of the Query.
③	**Description**	Remarks about the definition.

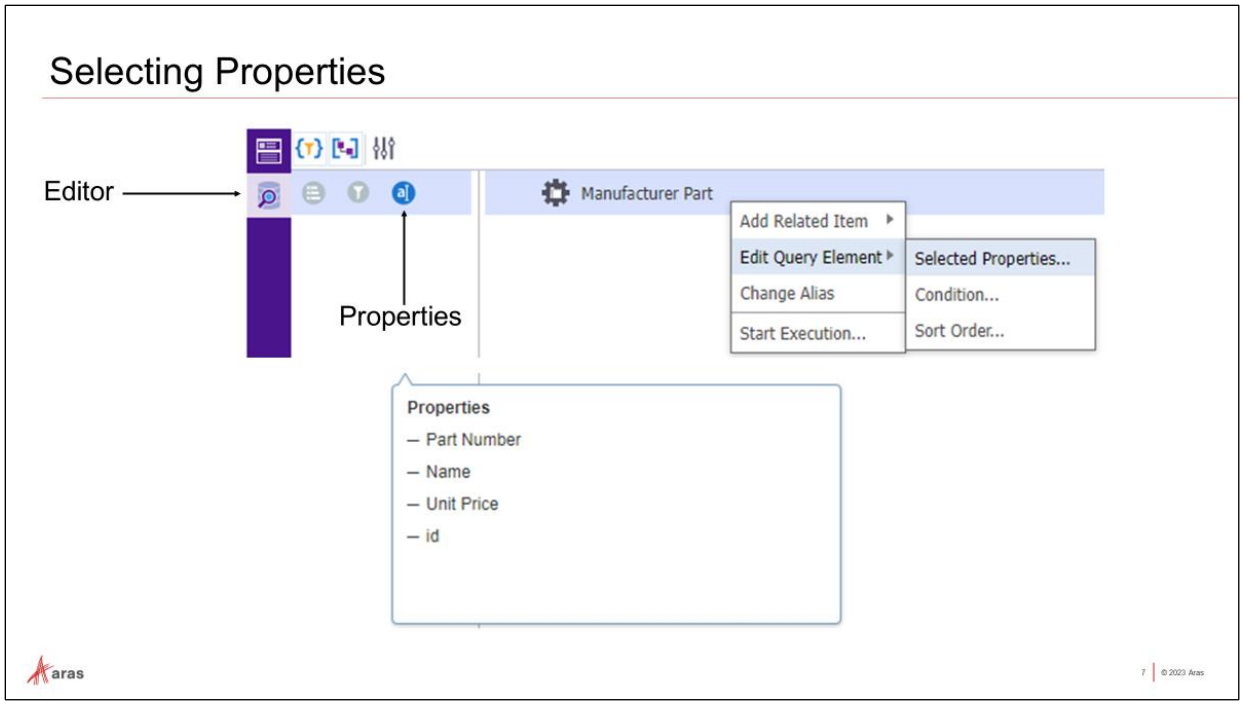
Try it … Create a New Query Definition

In this exercise, you create a Query Definition showing the pre-built relationships of Manufacturer Parts. This will be used later to display the information in a Tree Grid View.

1. Navigate to Administration > Configuration > Query Definitions and create a New Query Definition.
2. Enter the following:

Name	Manufacturer Parts with Relationships
Context Item Type	Manufacturer Part
Description	Manufacturer Parts and their associated data

3. Click Save.



Selecting Properties

After saving the Query Definition, you enter the query editing mode by clicking the Editor button in the sidebar. Then you can select the property values to be retrieved for each ItemType by using the Editor. You can add and remove properties at any time during the definition process.

Extended Properties (xProperties) and properties obtained from PolyItems are also supported. However, the xProperties must be assigned to the ItemType before you can use them.

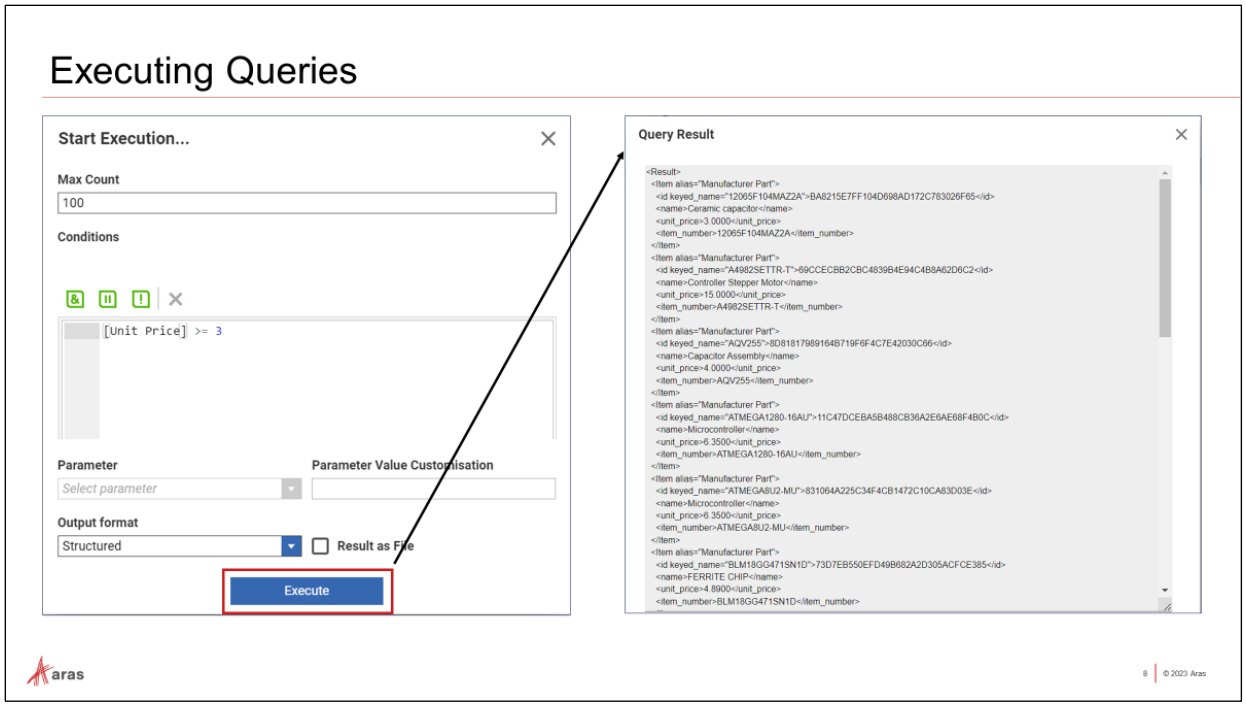
In this exercise, we define which Manufacturer Part property values we want to be returned.

Select an item in the right pane and use the context menu to choose properties from the Query Element or related Items and Relationships. Properties that were given a label on the ItemType will display the label in the selection window. Once selected, properties can be displayed and edited by hovering over the property icon in the left pane.

Try it … Select Properties

1. Click the Editor sidebar button.
2. Select the first row (Manufacturer Part) in the right-side pane, right-click and choose Edit Query Element > Selected Properties… from the context menu.
3. Use the Properties dialog to add the following properties to the query definition:

   Part Number, *Name*, *Unit Price* and *id*.
4. Click Save to store the values, which also closes the Query Element dialog box.
5. Click the Save button to save the Query Definition.



Executing Queries

You can test a query definition at any time by using the Execute Query Action. The AML results of the executed definition display in a Query Result window.

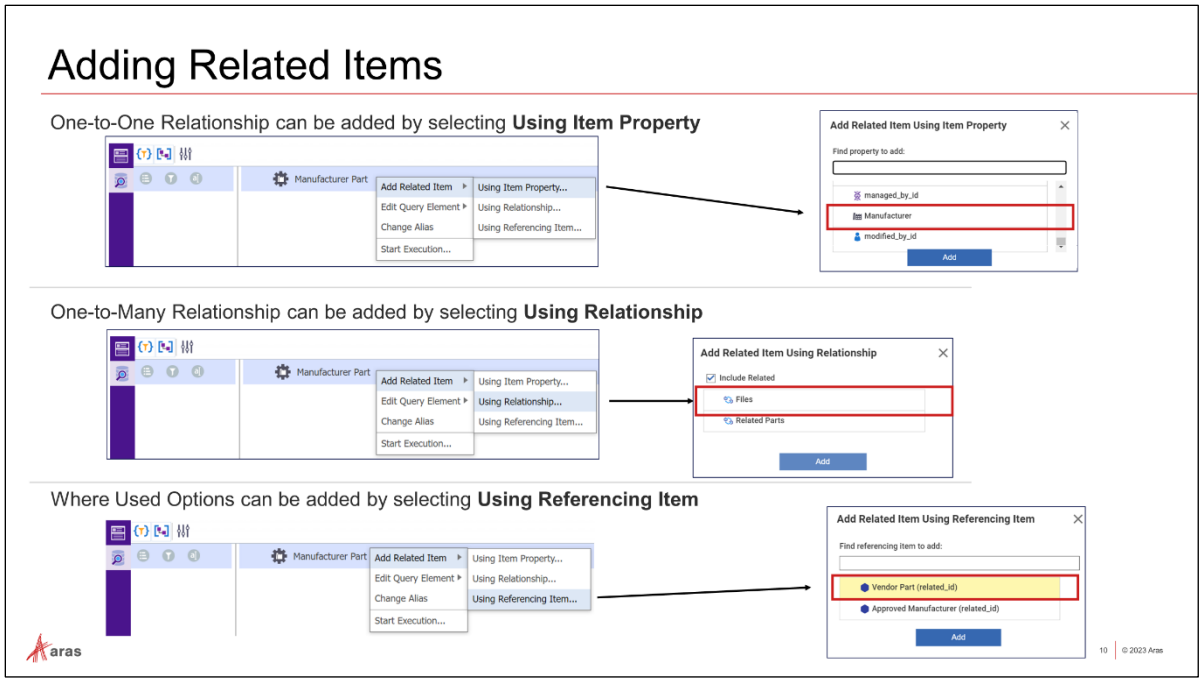
You can also specify Conditions and Parameters to filter the results. To supply a test condition, enter the name of a property in the Conditions editor and set a value. The editor auto-fill prompts for properties defined on the item. You must enclose Property names, including spaces, in brackets.

The Conditions editor supports basic SQL evaluation including =, >, <>, LIKE, IS NULL, etc. You can also create more complex statements using Boolean operators AND, OR, and NOT. Spaces must be inserted to separate properties, operators, and values. A default Max Count of 100 is displayed in the Start Execution window to limit the number of returned results.

You can save the query results as a file by selecting the Result as File checkbox before executing the query.

Try it … Execute a Query

1. Choose Execute Query from the More button.
2. In the Conditions field, enter the following value:
   [state]='Released'
3. Click the Execute button to run the query and display the results.
4. Close the Query Result window.



Adding Related Items

You can also add related data to a query. Right-click on the ItemType to select and use the Add Related Item option:

- *Using Item Property* to select a one-to-one relationship defined as an Item data type
- *Using Relationship* to select a one-to-many relationship added to the context ItemType
- *Using Referencing Item* to select options from a “Where Used” reverse lookup

When you make a selection, a dialog window opens that allows you to select from an existing relationship where the context ItemType is defined as the source of the relationship. You can remove a selection at any time by selecting a row in the editor and choosing **Remove Item** from the context menu.

In the next try-its, you enhance the query so that it fetches related items from a Manufacturer part: its Manufacturer, attached Files, its Vendors and related Parts; and include some properties from those related items.

Try it … Add an item from an Item Property and add Properties

1. Right-click the **Manufacturer Part** row (first row) and choose Add Related Item > Using Item Property from the context menu.
2. Select the **Manufacturer** item property, click the Add button to add the relationship to the query definition.
3. On the **Manufacturer** row (second row), click on the Properties icon on the left pane, and choose the following manufacturer properties: *Name, Country, Contact Name,* and *id.*
4. Click Save to store the values, which also closes the Query Element dialog box.

Try it … Add items from a Relationships tab and include properties

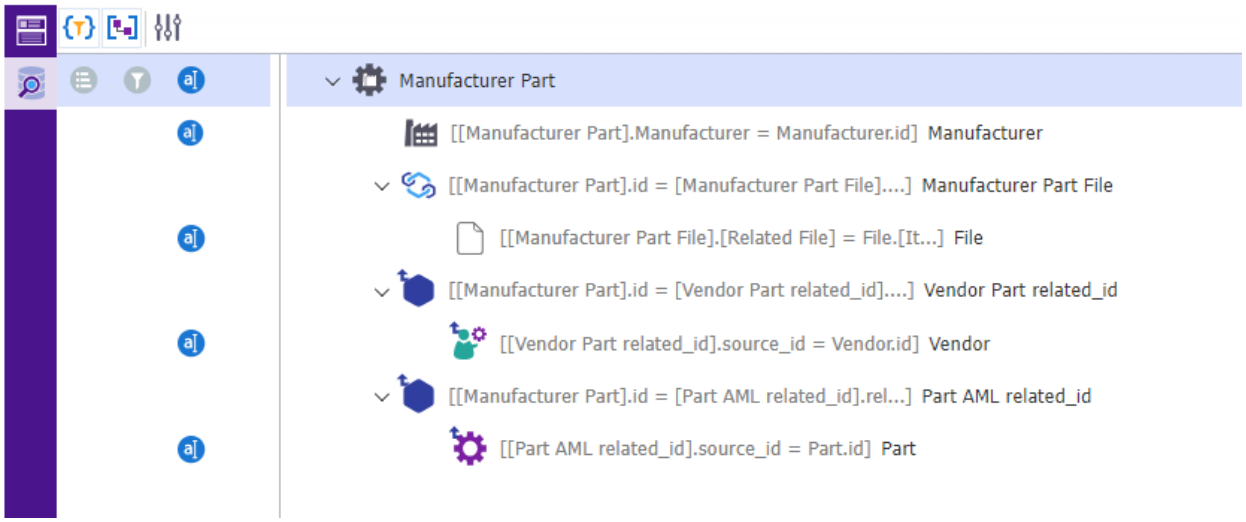
1. Right-click the **Manufacturer Part** row (first row) and choose Add Related Item > Using Relationship …

2. Select the **Files** relationship, then click Add.
3. Expand the relationship on the new third row.
4. On the File row (fourth row), click on the Properties icon in the left pane, and choose the following File properties: *File Name* and *Item ID* and Click Save.

Try it … Add Referenced Items and include properties

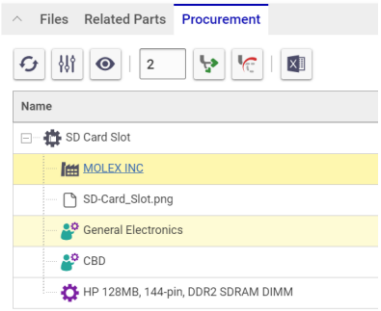
1. Right-click the **Manufacturer Part** row (first row) and choose Add Related Item > Using Referencing Item…
2. Select the *Vendor Part* relationship, then click Add.
3. Right-click on the new fifth row, then Add related item > Using Item Property, then select *source_id* property (with the vendor icon) and click Add.
4. On the **Vendor** row (sixth row), click on the Properties icon on the left pane, and choose the following Files properties: *Name, Country, Contact Name,* and *id* and Click Save.
5. Right-click the **Manufacturer Part** row (first row) and choose Add Related Item > Using Referencing Item…
6. Select the *Approved Manufacturer (related_id)* relationship and click Add.
7. Right-click on the new seventh row, then Add related item > Using Item Property, then select *source_id* property (with the part icon) and Click Add.
8. On the **Part** row (eighth row), click on the Properties icon on the left pane, and choose the following properties: *Part Number, Name, Cost,* and *id* and Click Save.
9. Click Done on the query definition.

The display of the Query editor should resemble the screenshot below.



Describing the Tree Grid View (TGV)

The TGV Shows Items *that reference* the context item and items *that are referenced* by the context Item.



Part Number	Name	Gen	Qty	Type
C4703A	DesignJet 2000CP Printer	2		Product
C4723-69096	Carriage assembly	1	1	Assembly
C4704-60117	Line sensor lens cover	1	1	Component
C6659-60190	Pen driver PC board	1	1	Component
0515-4700	Screw (Plastic) - M3x12....	1	1	Component
C6229A	4MB SIMM memory mo...	2	4	Component
C4704-60255	Front (Control) panel an...	1	1	Assembly
5063-1256	IEEE 1284 Bi-Tronics pa...	2	1	Component
C4704-60101	Front panel overlay	1	1	Component
C4723-69294	Firmware SIMM kit (Revi...	1	1	Component
C6659-60055	Power supply assembly	1	1	Assembly
RM1-4776-000...	Power supply	1	1	Component

aras

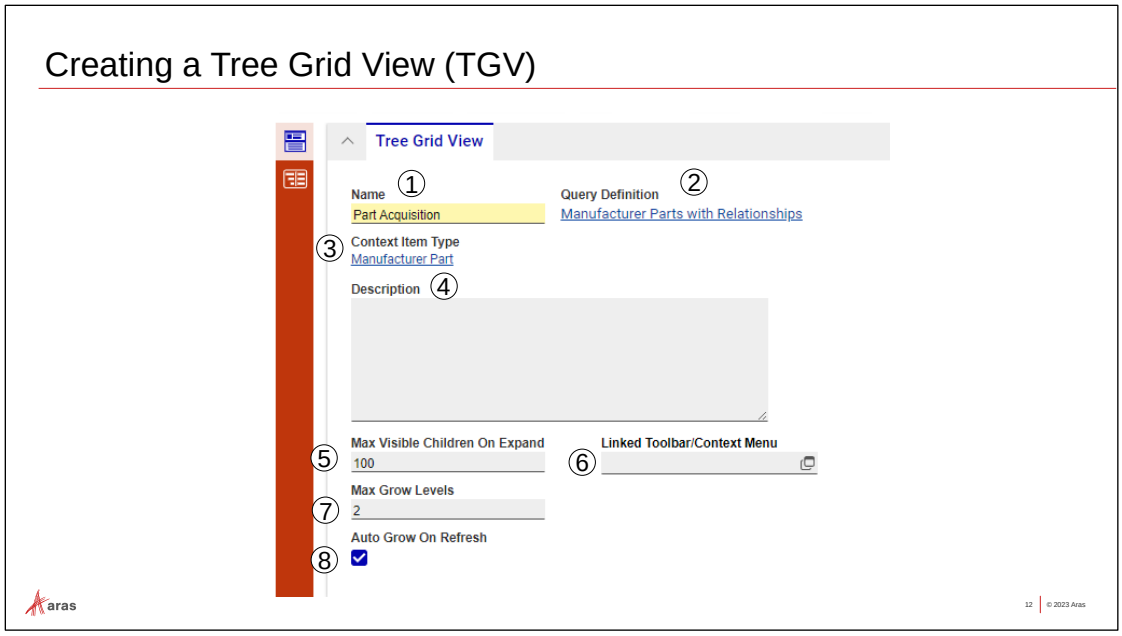
11 | © 2023 Aras

Describing the Tree Grid View

A Tree Grid View is a configurable user interface component that shows data using a hierarchical view. Administrators create the grids and choose which items and properties are located in each column and row. Users can expand and collapse the grid to show many kinds of items that are related to, or from, a starting context item.

You first create the Query Builder Definition to retrieve the desired items. You can then map the related data to a Tree Grid View to allow users to navigate easily through the returned results.

The query can go down through the defined child structure and up through the parent structure.



Creating a Tree Grid View (TGV)

You can use a Tree Grid View to display data that is retrieved from a Query Definition. The TGV can be associated with a relationship tab or with a custom client control (using JavaScript code).

The Tree Grid View form contains the following fields:

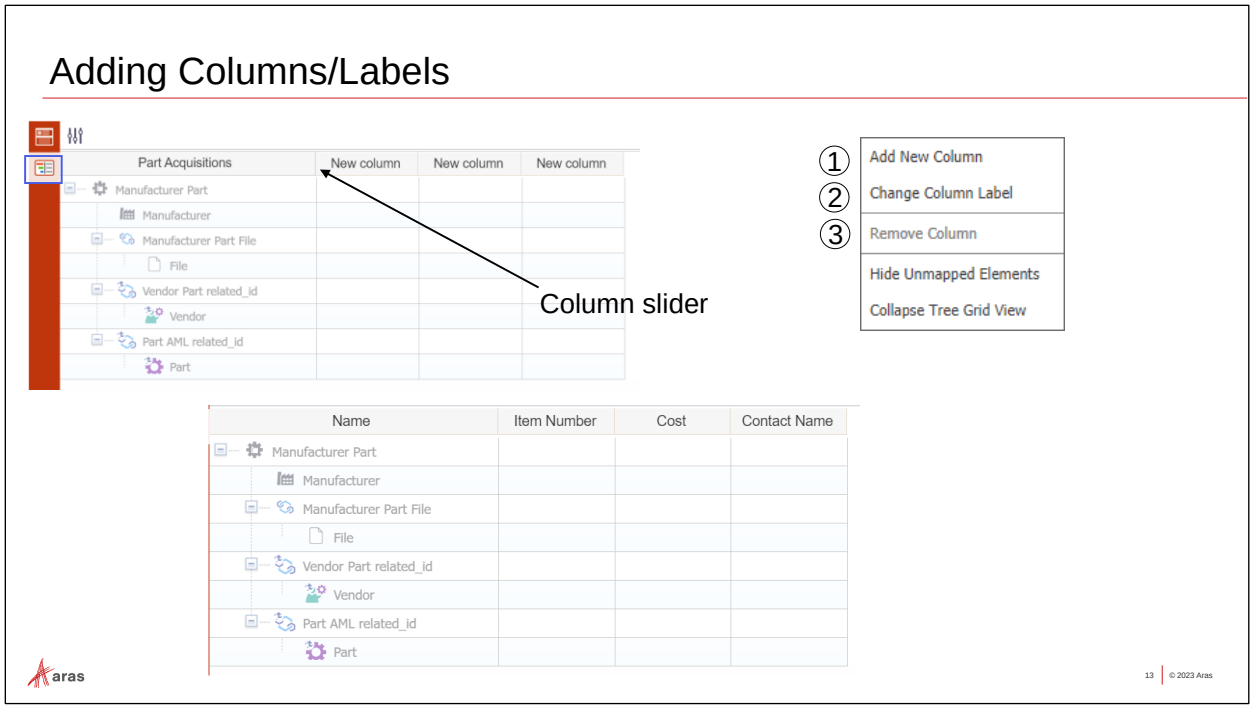
①	**Name**	The Tree Grid View name; max. length of 26 characters
②	**Query Definition**	A definition constructed using the Query Builder.
③	**Context Item Type**	The ItemType of the starting context item. This is provided by the selected Query Definition above.
④	**Description**	Comments about the Tree Grid View.
⑤	**Max Visible Children On Expand**	The maximum number of Children to display when the Tree Grid View is expanded.
⑥	**Linked Toolbar/Context Menu**	An optional Toolbar menu that has been created using the Configurable User Interface (CUI).
⑦	**Max Grow Levels**	The maximum number of levels to display when the Tree Grid View is expanded.
⑧	**Auto Grow On Refresh**	When selected, automatically grow the Tree Grid View upon clicking the Refresh button.

Try it … Create a Tree Grid View

10. Create a new Tree Grid View from Administration > Configuration > Tree Grid Views, as follows:

Name	Part Acquisition
Query Definition	Manufacturer Parts with Relationships
Auto Grow On Refresh	checked

11. Click Save.



Adding Columns/Labels

Data can be displayed to the user in a table format using a Tree Grid View. You can create additional columns to show the values of properties from the Query Definition. You decide which properties will display in each column and label each column header with appropriate text. You can adjust the width of each column by using the column slider.

The following options are available from the column context menu. (Must hover over column header)

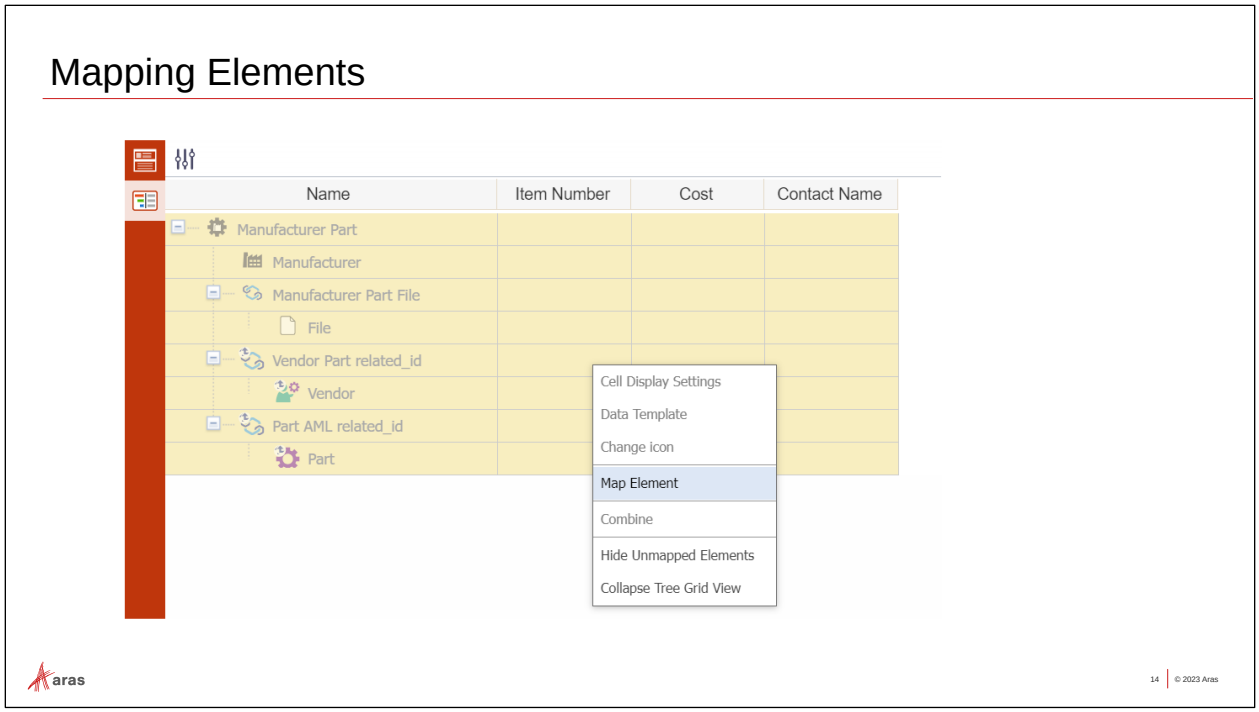
①	**Add New Column**	Inserts a new column to the right of the currently selected column.
②	**Change Column Label**	Allows editing of the column header text.
③	**Remove Column**	Removes the currently selected column from the TGV.

Try it … Add Columns/Labels

1. Click the Editor sidebar button to display the Tree Grid View Editor.
2. Right-click on the **Part Acquisition** column header and choose Add New Column from the context menu.
3. Add 2 more columns for a total of 4 columns (one existing and 3 new).
4. Label each column, as follows, by right-clicking on each column heading and selecting Change Column Label.

Name	**Item Number**	**Cost**	**Contact Name**

5. Save the changes.



Mapping Elements

Exposing property values in a Tree Grid View requires mapping an element from the Query Definition to the TGV control. All elements are unmapped by default. In the Tree Grid View Editor, select one or more rows to be mapped and then choose **Map Element** from the (right-click) context menu. To remove element mapping, select the desired rows and then choose **Unmap Element** from the context menu. Unmapped elements will not be shown in the TGV (e.g., in some circumstances you may want to hide the top row which is the context item, i.e. the root item).

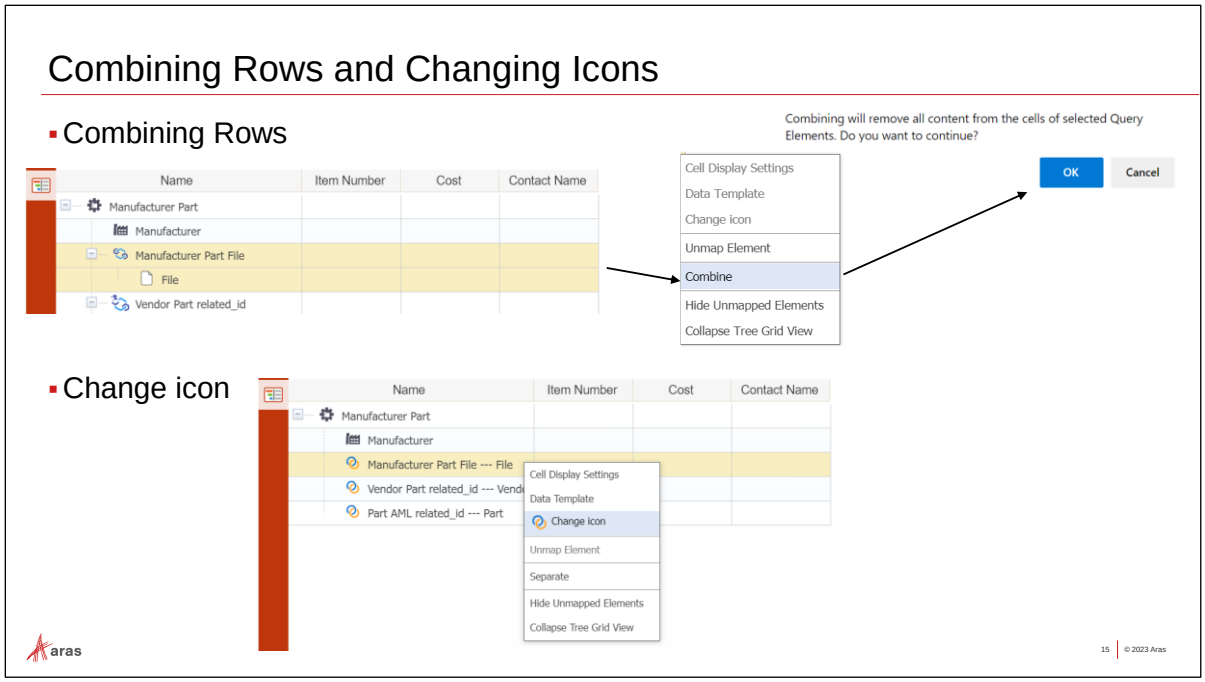
In the example above, the Query Definition retrieves items related to a manufacturer part.

To suppress elements that will not be mapped in the editor, choose Hide Unmapped Elements. For larger Query Definitions this can make property mapping easier by eliminating rows that will not be used.

Try it … Map Elements

Use this exercise to map all rows of the Query.

1. Select all rows, right-click, and select **Map Element**.
2. Save the Tree Grid View.



Combining Rows and Changing Icons

The Tree Grid View contains common configurable features to enhance the display and usability of the view. One common configurable feature allows Query Definition rows to be combined so that property data from more than one item can be displayed on a single row. To combine rows, use the Ctrl key to multi-select the rows to be combined, right-click, and choose Combine from the context menu.

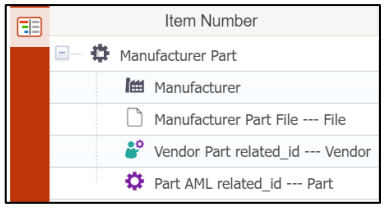
When you combine rows, any settings you previously configured for displaying the properties in the cells will be removed. You must acknowledge a warning dialog to continue. You will configure the cell display settings after combining the rows.

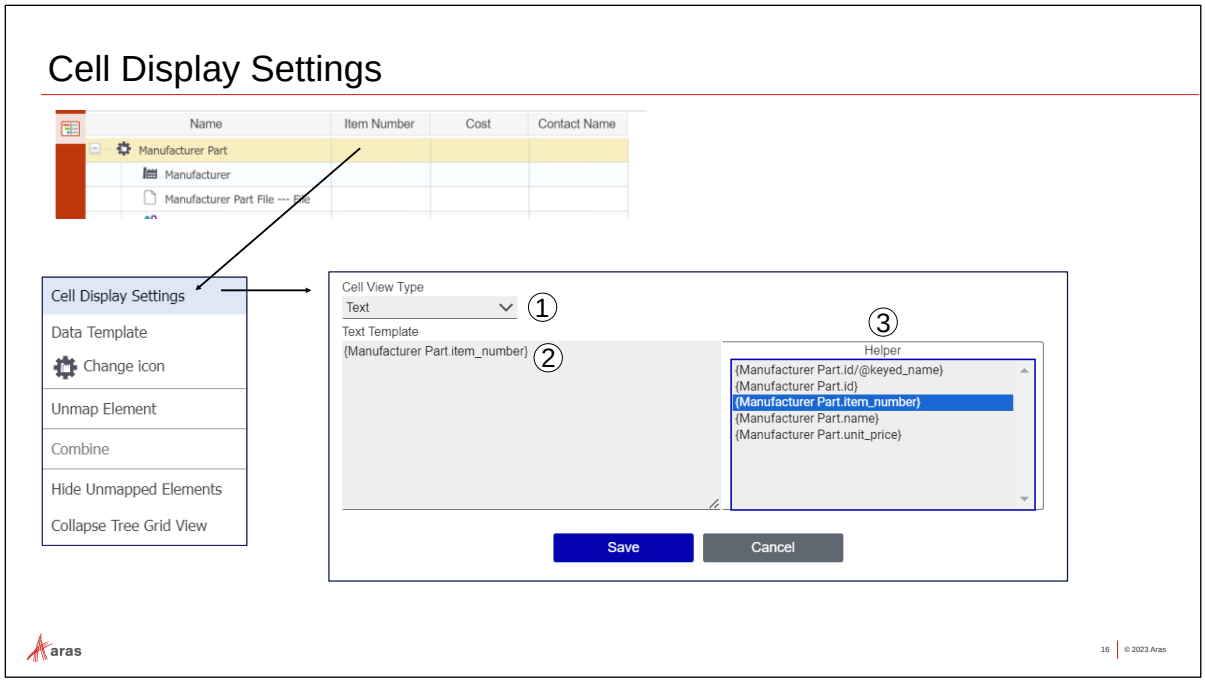
Try it … Combine Rows

1. Combine both the Manufacturer Part File element (row 3) and the related File element (row 4) by selecting both rows, right-click, and choose Combine.
2. Select OK in the pop-up warning to continue.
3. Repeat those steps to combine the *Vendor Part related_id* with the *Vendor* and the *Part AML related_id* with *Part*.
4. Save the Tree Grid View.

Try it … Change the Icon

1. Right-click on the Manufacturer Part File --- File (third) row and choose Change icon.
2. In the resulting Image browser window, select Images.
3. Since this row represents the Children Files, find and select the **File** icon.
4. Repeat the steps above to select appropriate icons for the last two rows.
5. Save the Tree Grid View.





Cell Display Settings

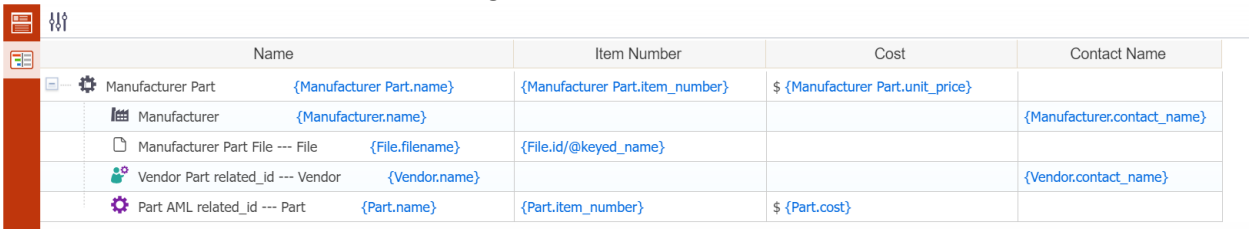
Once a query element (row) is mapped, the properties of that element can be selected from the Query Definition to display in the Tree Grid View. Use the Cell Display Settings editor to provide instructions about what is displayed based on the Cell View Type selected. To configure a display setting for a cell, click to select a cell, and choose Cell Display Settings from the context menu.

The Cell Display Settings editor contains the following fields:

①	**Cell View Type**	The Data type for cell display: choices include Text, Decimal, Date, Color, Item, List, Float, Boolean, and Integer.
②	**Text Template**	Text and property data to be displayed in the current cell. Properties are always enclosed in curly brace characters {} and use the syntax **ItemType.propertyname**. You can provide more than one property expression in a cell as well as entering free form text. For example, you can add a $ character before a property.
③	**Helper**	The Helper window shows you the properties that have been configured in the Query Definition for the currently mapped element. Double click on a property to add it to the Text Template window.

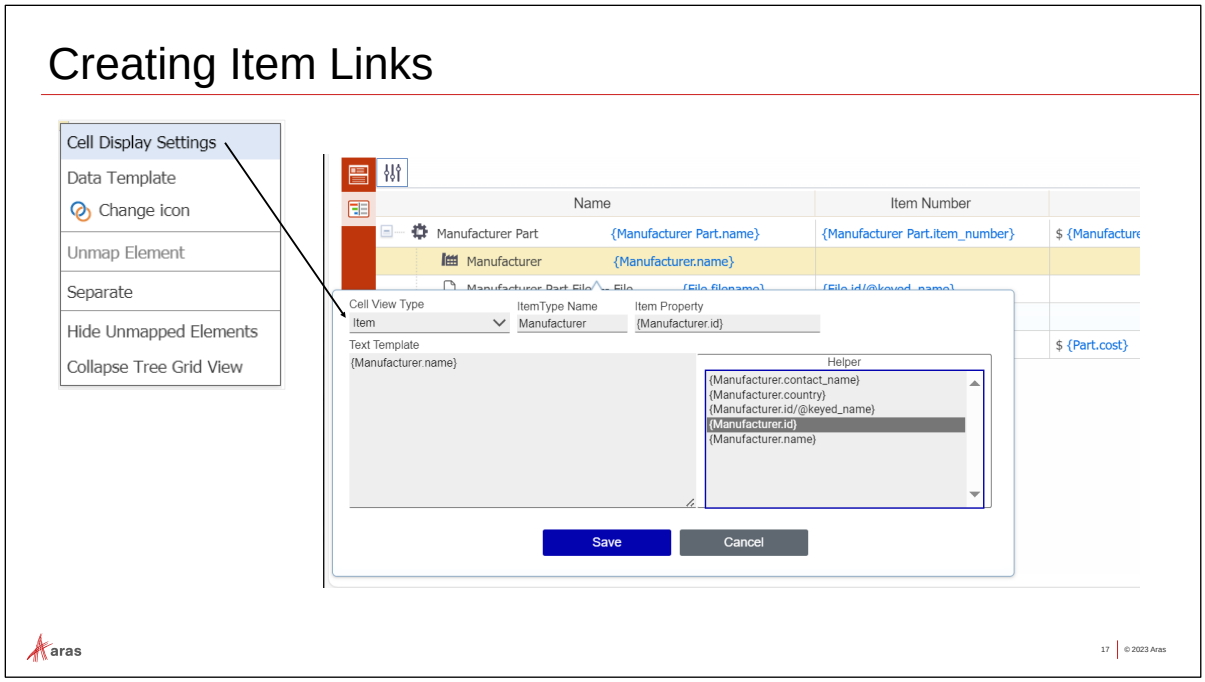
Try it … Configure Cell Display Settings

1. Select the first cell of the first row under the Name column, right-click, and choose Cell Display Settings to open the Cell Display Settings dialog window.
2. Double-click the **{Manufacturer Part.name}** property in the Helper section to move into the Text Template section. If you make a mistake, just select, and delete from Template section.
3. Click Save to close the Cell Display Settings dialog window.
4. Repeat by selecting the top cell under Item Number column and map **{Manufacturer Part.item_number}** to that cell.
5. Repeat by selecting the top cell under Cost column and map **{Manufacturer Part.unit_price}** to that cell. Add a *$* and space in front of the property in the Text Template.
6. Repeat the previous steps on the cell in the first column on the second row, to add **{Manufacturer.name}**.
7. Repeat the previous steps to select the cell under Contact Name for the Manufacturer row, to add **{Manufacturer.contact_name}**
8. Repeat the steps to add **{File.filename}** for the cell Under Name column on the Manufacturer Part File-File row.
9. Repeat the steps to add **{File.id/@keyed_name}** for the cell Item Number on the Manufacturer Part File-File row.
10. Configure cell settings to add **{Vendor.name}** and **{Vendor.contact_name}** to the Vendor row.
11. Continue to edit the cell display settings of the related Part row (Referenced Item) by adding **{Part.name}**, **{Part.item_number}** and $ **{Part.cost}** in the appropriate columns.
12. The TGV editor should show the following result:



Name		Item Number	Cost	Contact Name
Manufacturer Part	{Manufacturer Part.name}	{Manufacturer Part.item_number}	$ {Manufacturer Part.unit_price}	
Manufacturer	{Manufacturer.name}			{Manufacturer.contact_name}
Manufacturer Part File --- File	{File.filename}	{File.id/@keyed_name}		
Vendor Part related_id --- Vendor	{Vendor.name}			{Vendor.contact_name}
Part AML related_id --- Part	{Part.name}	{Part.item_number}	$ {Part.cost}	

13. Save the Tree Grid View.



Creating Item Links

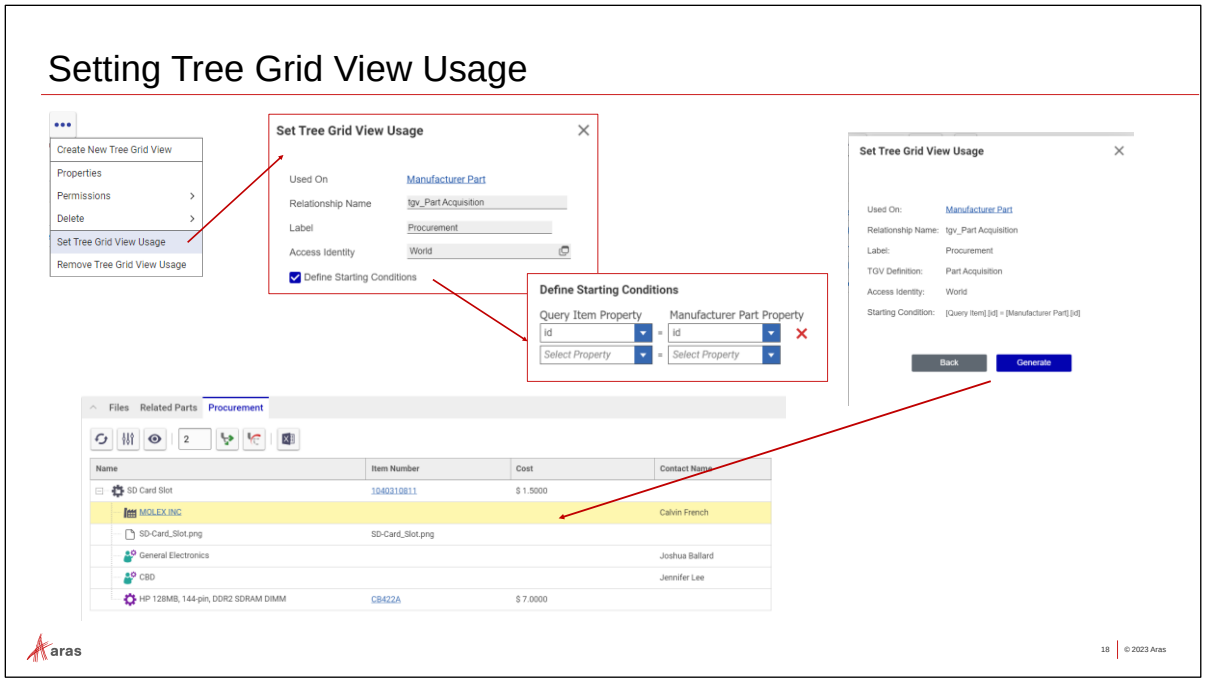
You can enhance the Tree Grid View by converting a property value to a hyperlink. The user will be able to click the link to open the item. You have edited the cells to include the text version of whatever the property was set to. A new Item cell view type will have to be added to the cells already containing the text template to transform to a link which will open an item that exists within the database.

After selecting **Item** from the Cell View Type, two new fields will appear. The ItemType and the id property must be added to create the hyperlink before clicking save. You must type in the ItemType Name. Next, move the insertion point to the Item Property field and double-click from the Helper window to select the id property.

Try it … Create Item Hyperlinks

Now let's enhance the TGV by adding a hyperlink to the related items.

1. Open Cell Display Settings for **Manufacturer** (second row)
2. Choose **Item** from the Cell View Type dropdown list.
3. For the ItemType Name field, enter *Manufacturer*.
4. Click in the Item Property field and then double-click the **{Manufacturer.id}** value in the Helper section to select it as the hyperlink destination as shown above.
5. Repeat the steps above to convert the Part Item Number into a hyperlink in the second column.
6. For the ItemType Name field, enter *Part*.
7. Select **{Part.id}** as the Item Property.
8. Click the Save button and then save the Tree Grid View.



Setting Tree Grid View Usage

Once a Tree Grid View has been configured, it is then assigned to an ItemType or a custom JavaScript Method. The Set Tree Grid View Usage wizard steps you through the process of making the assignment to generate the configuration.

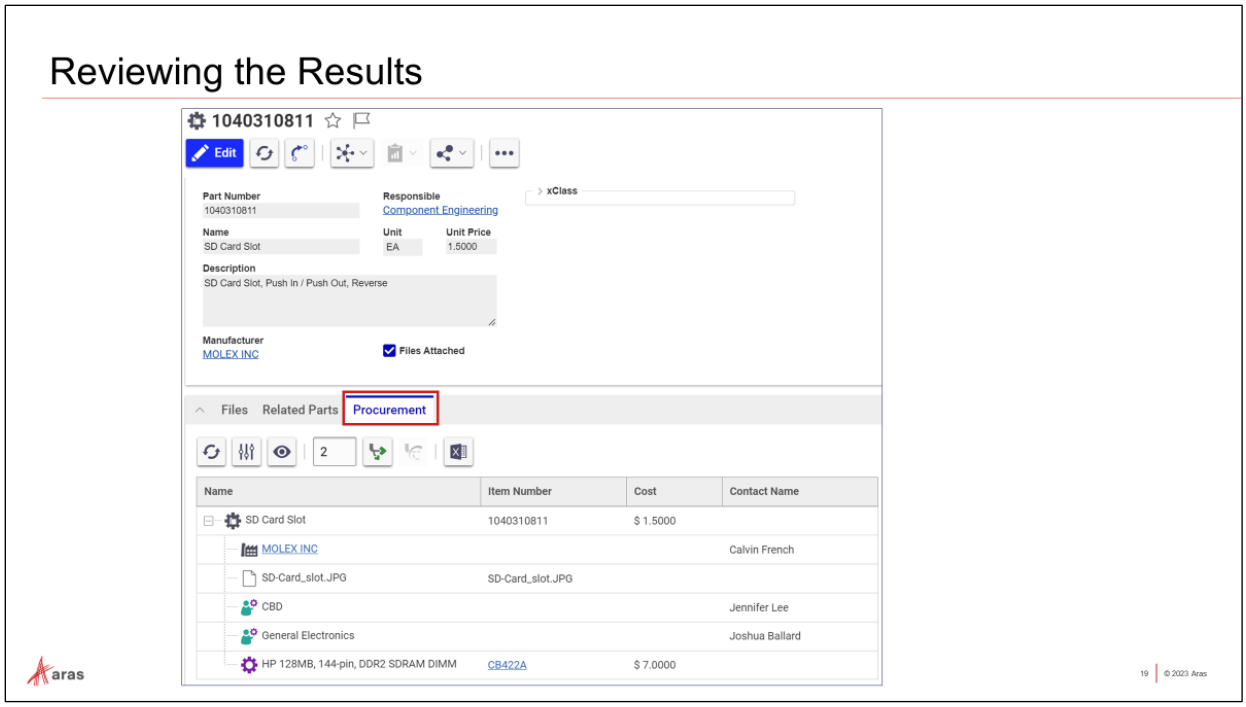
The Define Starting Conditions screen allows you to provide starting property values to the underlying Query Definition. The summary screen shows the selections you have made. To make changes you would use the Back button.

To remove a Tree Grid View from the targeted ItemType, use the Remove Tree Grid View Usage action in the More menu. A list of current relationships is presented that use the view. Select the relationship(s) to be deleted and click the Remove button.

Try it … Set TGV Usage

In this exercise, you run a wizard to create a new Relationship tab for our TGV.

1. Open the More… menu and select Set Tree Grid View Usage.
2. Select *Relationship Tab* for Target Usage and click Next.
3. Ensure the targeted Item Type (Used On) is **Manufacturer Part** and ensure that New (Relationship Name) is selected, then click Next.
4. A default name for the relationship is provided. For the Label field, enter *Procurement*, use the **World** identity, leave the Define Starting Conditions checkbox selected, and click Next.
5. Leave the **id** of the current Manufacturer Part item to populate the Procurement tab and click Next.
6. Click Generate to create the new relationship (and its associated new tab) on the targeted ItemType.
7. When you see the Configuration Generated Successfully Screen, click Done.
8. Now click Done for the TGV.



Reviewing the Results

To display a Tree Grid View, open an item based on the targeted ItemType and select the assigned relationship tab that corresponds to the Tree Grid View.

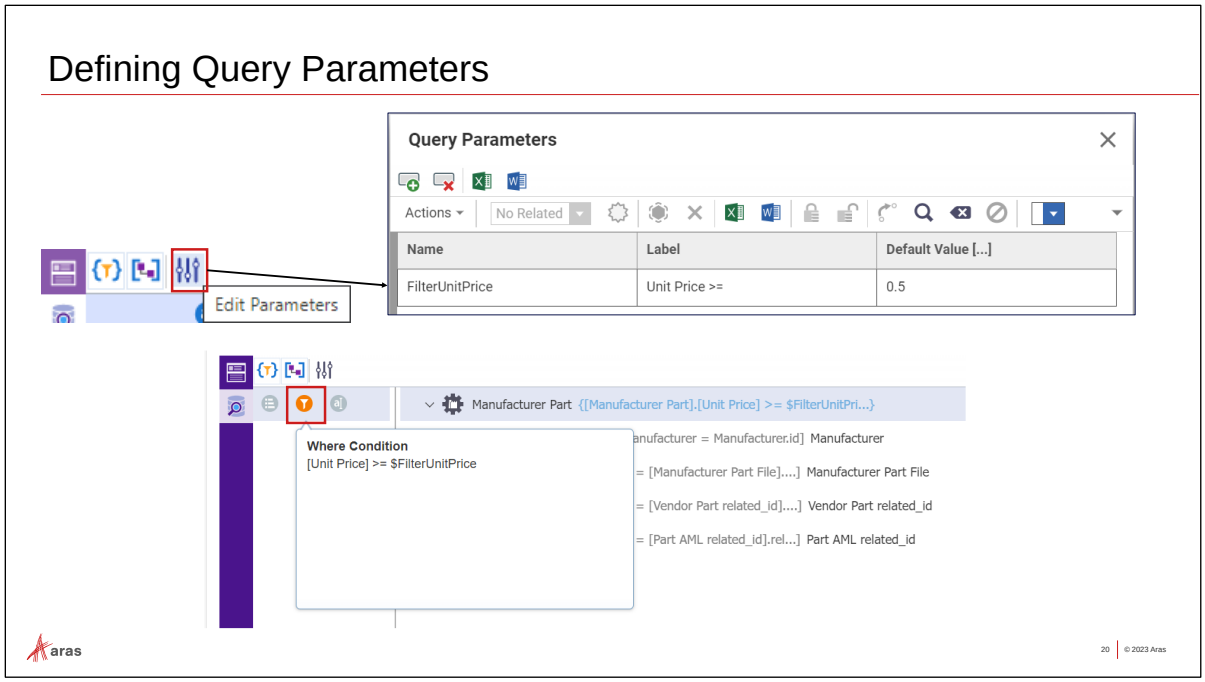
Try it … Review the TGV Result

In this exercise, we complement the dataset and observe the display on the TGV.

1. Navigate to Sourcing > Vendors. Create a new Vendor as follows:

Name	Newtex
Main Phone	(718) 410-2827
Contact Name	Jennifer Walden

2. In the Vendor's relationship tab, add the Manufacturer part **93395A213**. Click Done.
3. Navigate to Sourcing > Manufacturer Parts; open for editing the manufacturer part **93395A213**. Add a New file to the Files Relationship tab.
4. Name this Manufacturer Part *Steel Flat-Head Socket Cap* and set the Unit Price to 1.5. Click Save.
5. Select the *Procurement* tab. The results from the Query are displayed in the new Tree Grid View. The tree grew automatically from the default settings, and displays the relationships of 93395A213. Hyperlinks are available as defined.
6. Click on some of the hyperlinks to confirm that the linked items open in a new tab.
7. After viewing the items from Procurement tab, click Done and close the Manufacturer Part **93395A213** and all related items.



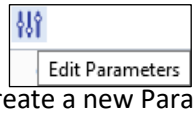
Defining Query Parameters

Administrators create Parameters for Query Definitions and use them as variables in a Condition statement. Users can then supply criteria to filter the results of a query. The Parameter is created first and then used to create a Where Condition. The Filter Condition icon becomes orange when a condition is applied to the row. Conditions support operators, but spaces must separate operators and properties.

Try it … Define a Query Parameter

In this exercise, we will add a filter to the Query Definition previously created.

1. Open the **Manufacturer Parts with Relationships** Query Definition for editing and click the Editor button on the sidebar.
2. On the Query Editor toolbar, click the Edit Parameters button.

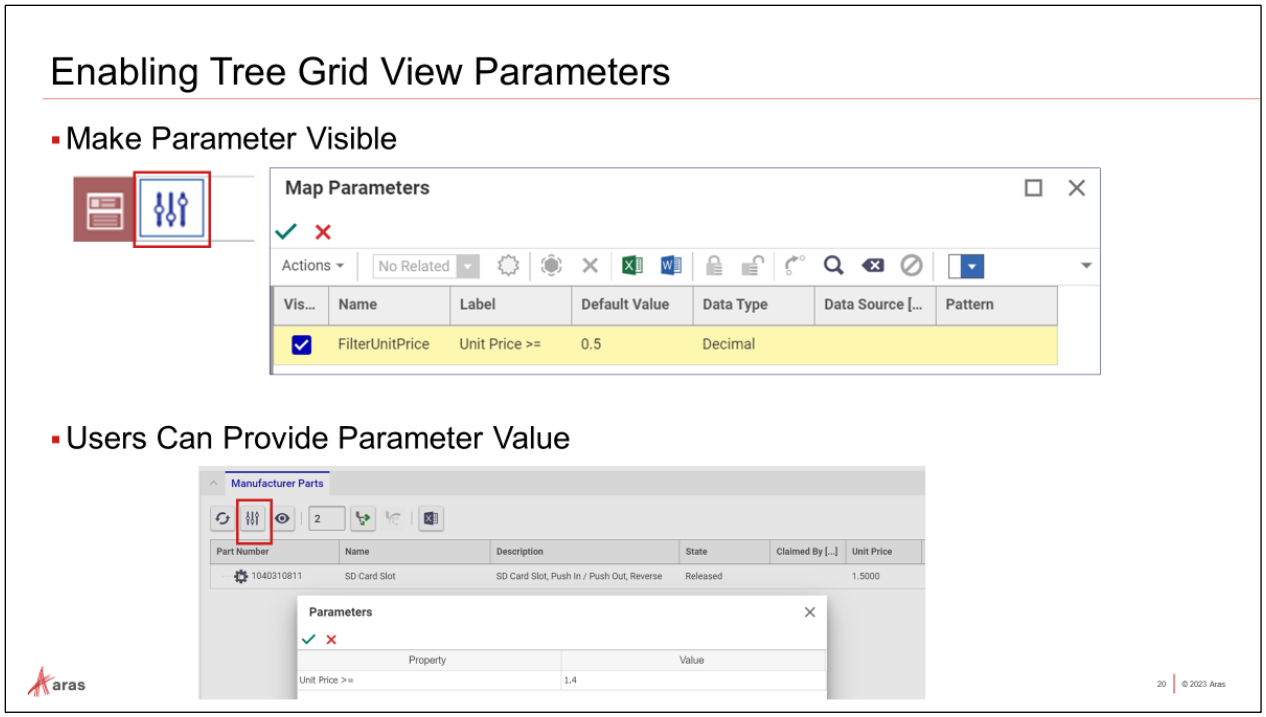


3. In the Query parameters window, click the New (+) button to create a new Parameter.
4. Name the Parameter *FilterUnitPrice* (no spaces allowed) and enter *Unit Price >=* for Label.
5. Click in the Default Value cell, and in the Text Dialog window, enter a value of *0.*5 and click Save.
6. Close the Query Parameters window (by clicking the X) to accept the values, then save the Query Definition.

Try it … Map a Query Element to a Condition

In this exercise, you map a Query Element to a Where Condition. To use parameters in a Condition statement, prefix the Parameter Name with a $ (dollar sign).

1. With the **Manufacturer Parts with Relationships** Query Definition still open for editing, on the first row, click the Filter icon in the left panel to open the Where Condition dialog.
2. Enter *([Unit Price] >= $FilterUnitPrice).* Notice that the properties with a space, such as ‘Unit Price’, require brackets.
3. Save the Where Condition and click Done on the Query Definition.



Enabling Tree Grid View Parameters

To allow users to supply parameter values when a Tree Grid View displays, you must configure the Parameter to be visible. The Map Parameters dialog window displays a list of parameters.

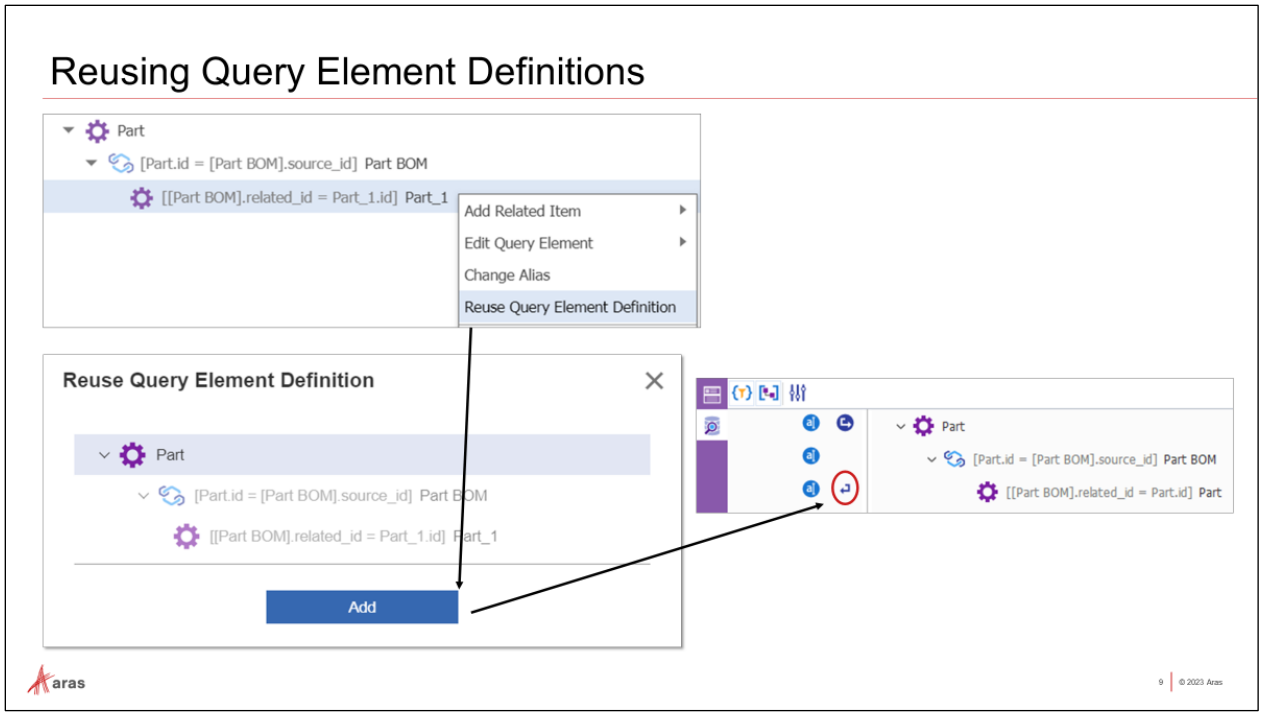
Once a Tree Grid View is configured with Parameters marked as visible, a user can click the Modify Parameter button in the TGV Relationship grid to supply new filter values. The grid view is updated when the user confirms the new value.

Try it … Enable a Tree Grid View Parameter

In this exercise, we will enable the Unit Price filter parameter to be used.

1. Open the **Part Acquisition** Tree Grid View for editing.
2. Using the Editor toolbar, click the Show Parameter Mapping button.

3. Click to select the Visible checkbox on the FilterUnitPrice row.
4. Choose **Decimal** for the Data Type.
5. Click the Save (green checkmark) button to accept the values and close the Map Parameters window, then click Done for the Tree Grid View.
6. Find and open the Manufacturer Part **93395A213**. Verify that you see this part as top element on the Procurement relationship tab.
7. Test the parameter by clicking the Parameters button and setting a different value greater than $1.50.
   Notice that the grid will display no results. The context item selected does not match the specified criteria, therefore none of the referenced items are displayed.
8. Test again by setting a value less than $1.50. The top element and relationships will display.



Reusing Query Element Definitions

A powerful feature of the Query Builder is the ability to perform recursion over a collection of data that is nested in the database. For example, the Bill of Materials for an assembly contains a list of parts which may in fact be assemblies that represent a further collection of parts, etc.

The Reuse Query Element Definition feature allows you to automatically perform recursion over a collection of data by reusing the same query definition repeatedly until no more data is found. Reuse also means that you are reusing the assigned properties. The Reuse icon is visible to the left of the selected row when the feature is enabled.

Try it … View the Reuse Query Element Definition

In this exercise, we examine the existing PE_BomStructure Query Definition.

1. Find and open the **PE_BomStructure** Query Definition.
2. Click on the Editor sidebar button and expand the hierarchical display.
3. Hover over the Properties icon in the left pane to see the list of properties for the top element. Notice there is an icon to the right to reflect that these properties will be reused in this query definition.
4. Navigate to the third row.
5. Hover over the Properties icon in the left pane to see the list of properties.
6. Notice that the properties are the same as the first row and there is a visible icon to the right showing a reuse of properties from a previous element.
7. Close the PE_BomStructure tab.

Summary

In this unit, you learned how to create a Query Definition that populates a Tree Grid View control.

You should now be able to:

- Review Existing Data Tools
- Describe the Tree View Grid
- Describe the Query Builder
- Create a new Query Definition
- Create a Tree Grid View
- Assign Tree Grid View Usage
- Enhance the Tree Grid View Display
- Enable Query Parameters

Review Questions

1. The Query builder creates reusable Query Definitions to retrieve data from the Aras Innovator database.
   What does the Query Definition use to return related items?
   What does the Query Definition use to return referencing items?

2. In which format does the Query returns data?

3. Since users often want to see data returned from the database displayed in columns and rows, what can you create to show results of a Query in a table format?

4. What must you do to elements of a Query Definition to expose property values in a Tree Grid View?

5. An end user can influence the results displayed by setting a parameter value. As an administrator, what do you need to configure to provide parameters?

6. How can you enhance a Tree Grid View to allow a property value to become a hyperlink to another item?

Lab Exercise

Goal

Be able to create a new Query Definition that references relationship and item property data and map the query results to a Tree Grid View control.

Scenario

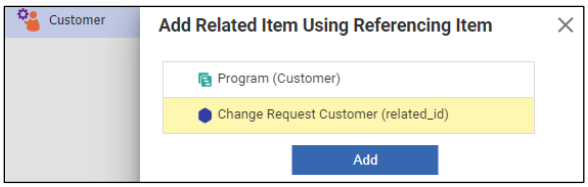
In this exercise, you will create a Query Definition that retrieves all referencing Change Requests for a Customer and add a Customer Change Requests tab that displays the results.

Steps

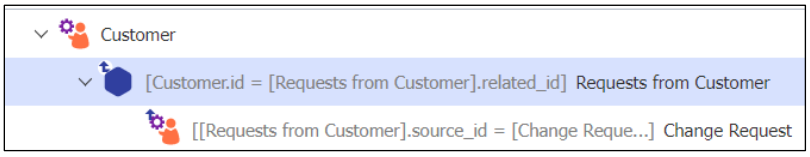
Ensure you are logged into Aras Innovator as the Admin user.

Create the Query Definition

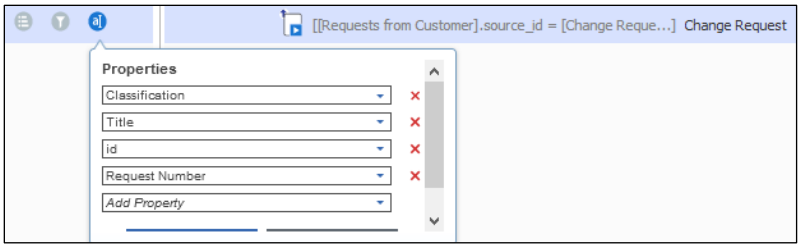
1. Create a new Query Definition item named *Customer Requests*, choose **Customer** as the Context ItemType, enter a description, and save the changes.
2. Click the Editor sidebar button to begin to build the definition.
3. Select the first row (Customer) in the right-side pane, right-click and choose Edit Query Element > Selected Properties… from the context menu.
4. Use the Properties dialog to add the following properties to the query definition:
   Name and *Main Phone*.
5. Right-click on the **Customer** ItemType (first row) and select Add Related Item > Using Referencing Item from the context menu.
6. Select **Change Request Customer (related_id)** from the Add Related Item Using Referencing Item dialog window and click Add.



7. To avoid a SQL error due to name length, on the Change Request Customer relationship (row 2), select Change Alias from the context menu, enter *Requests from Customer* as the new alias, and click the green checkmark to save the change.
8. From the Change Request Customer relationship (row 2: **Requests from Customer**), select Add Related Item > Using Item Property from the context menu and choose the **source_id** property, then click Add.

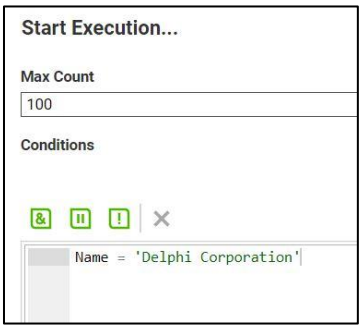


9. Select the Change Request element (row 3), and then add the following properties to the query definition: **Request Number, Classification, Title,** and **id**.



10. Save the changes to the properties window.
11. Save changes to the Query Definition, select Execute Query from the More menu, and execute the query.
12. To see requests for only a single Customer, first ensure you have created a CR for a specific customer.
13. Execute the query using that customer as a condition.

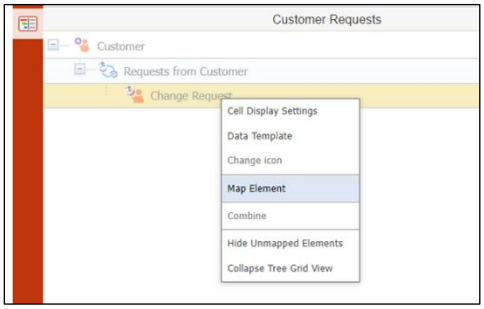
    This example uses Customer **Delphi Corporation**.
    Name = 'Delphi Corporation'



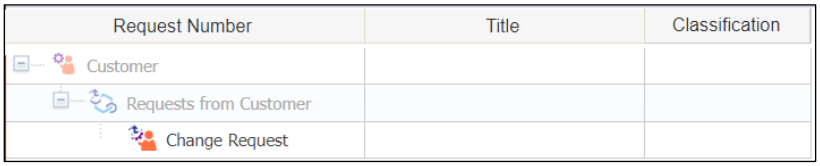
14. Click Done on the Query Definition.

Create the Tree Grid View (TGV)

1. Create a new Tree Grid View named *Customer Requests* using the new **Customer Requests** Query Definition, enter a description, and save the changes.
2. On the sidebar, click the Editor button, right-click on just the Change Request row (3) in the grid, and select Map Element.

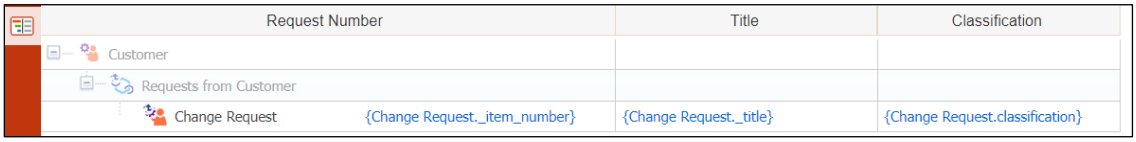


3. Add two new columns and change the column labels to show the Change Request Item Number **(Request Number), Title,** and **Classification** as shown here:

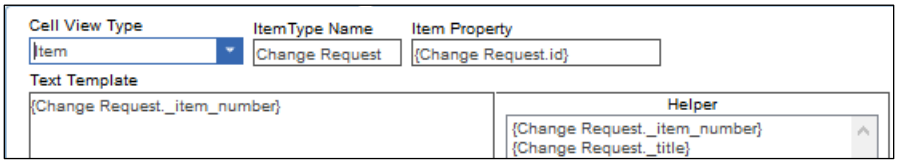


Configure the Cell Display Setting

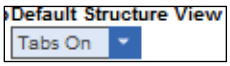
1. Configure the Cell Display Setting to show the **Change Request Item Number, Change Request Title,** and **Change Request Classification** as shown here:



2. Make the **Change Request Item Number** a hyperlink to the referenced Change Request item.



3. Click Save in the cell Display Settings dialog window.
4. Save the changes and then select Set Tree Grid View Usage from the More menu.
5. In the Set Tree Grid View Usage dialog window, select Relationship tab, and click Next.
6. Accept the **Customer** value for the Used On field, ensure the Relationship Name is set to New, and click Next.
7. For the Label field, you can enter: Change Requests.
8. *For the Access identity,* you can enter: Component Engineering.
9. Click next, accept the Starting Conditions by clicking Next, and click Generate.
10. When the TGV is generated, click Done in the dialog window, then click Done on the TGV.
11. Before you can test the results, you must open the **Customer** ItemType for editing, set the Default Structure View setting to Tabs On, and click Done.



Test the Query Definition and the TGV

1. Edit some existing Change Requests, replacing the related Customer to all be the same customer.
2. Also, ensure that each CR has a Classification (Defect or Enhancement) assigned.
3. Navigate to Portfolio > Search Customers, open for viewing the Customer you added to the Change Requests, and view the results on the *Change Requests* tab.

Page left intentionally blank.

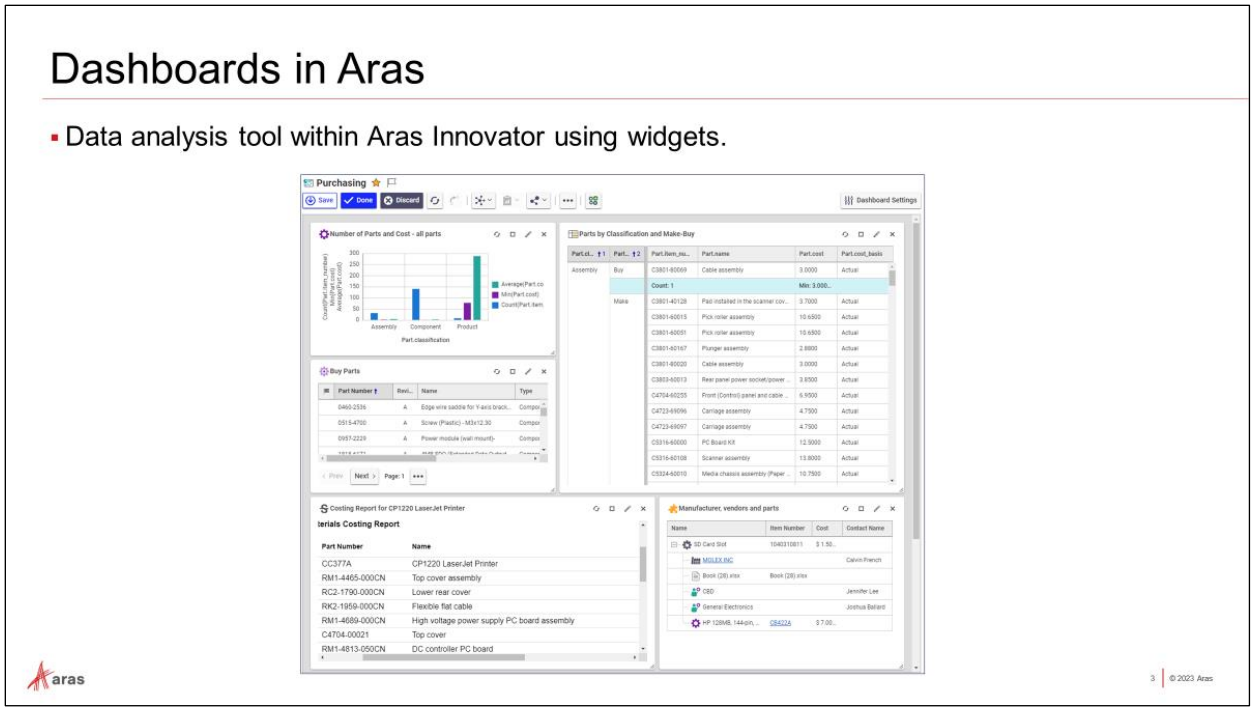
Unit 11 Dashboards and Reporting

Overview

In this unit, you will learn how to use Dashboards and their widgets, as well as Report items to present Aras data for data analysis in a user-friendly format. A knowledge of Query Definitions and Tree Grid Views is advised.

Objectives

- Manage Dashboards and widgets
- Understand the Dashboard security access
- Create, move, and delete widgets
- Manage Reports
- Export a report preview
- Create Report tables and report charts
- Manage a widget template

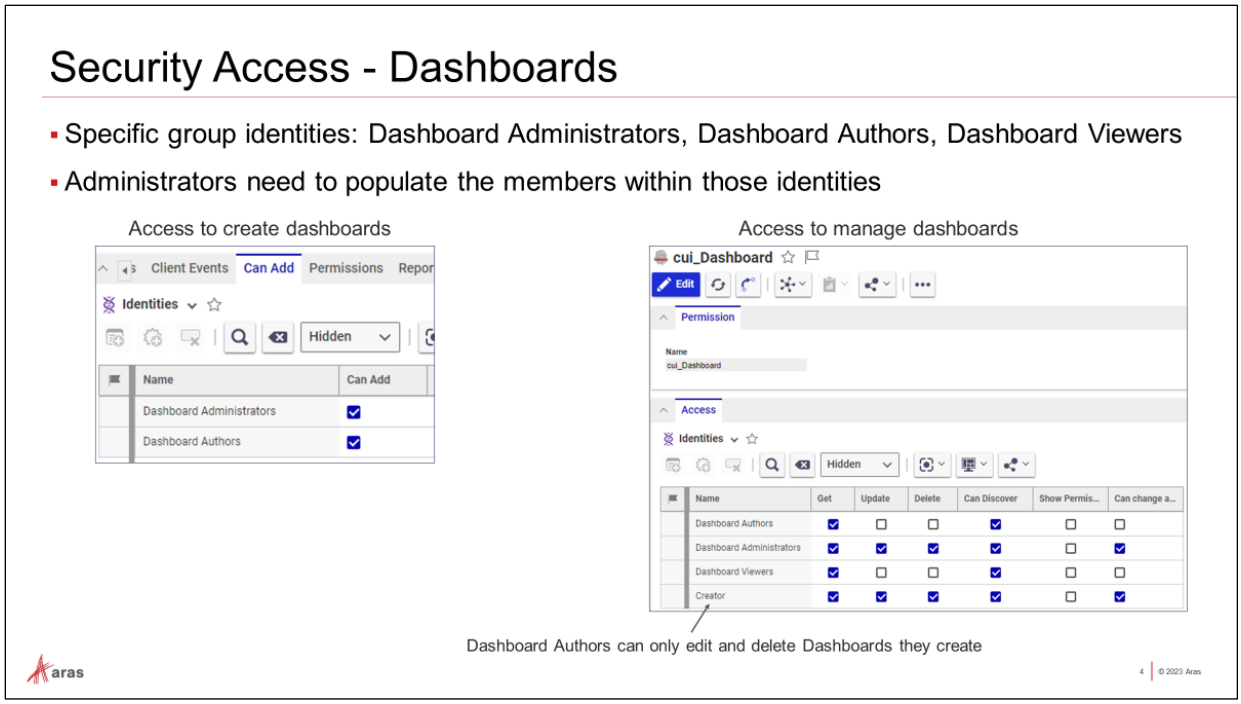


Dashboards in Aras

Dashboards are used to display Aras data in either real time or with a data snapshot. Each dashboard contains several interactive Widgets that can be of different types, including data grids, charts, reports.

The Dashboards have specific permissions to grant capacity of data analysis to specific groups in the company. Several Dashboards can be created for different purposes, see some examples below:

- Engineering:
  - Engineering Change Reports (New vs Closed, Pipeline, Days To Complete)
  - Parts And Assemblies (Released, Ready To Release)
  - OCM-User Adoption (Users Registered, Trained, Active, Not Active)
- Purchasing:
  - Costs rollup
  - Late Deliveries
  - Orders Not Placed
- Enterprise Master data management:
  - Part Global Attribute Mismatch (Desc, Rev, UOM, State)
  - BOM Accuracy (Parents Missing Children, Children Missing Parents, Qty)



Security Access - Dashboards

Specific group identities were introduced to manage Dashboards. The Aras Innovator administrator will need to populate the members within those identities.

Dashboard Administrators can configure all Dashboard Features, as well as create, edit, and delete Dashboards.

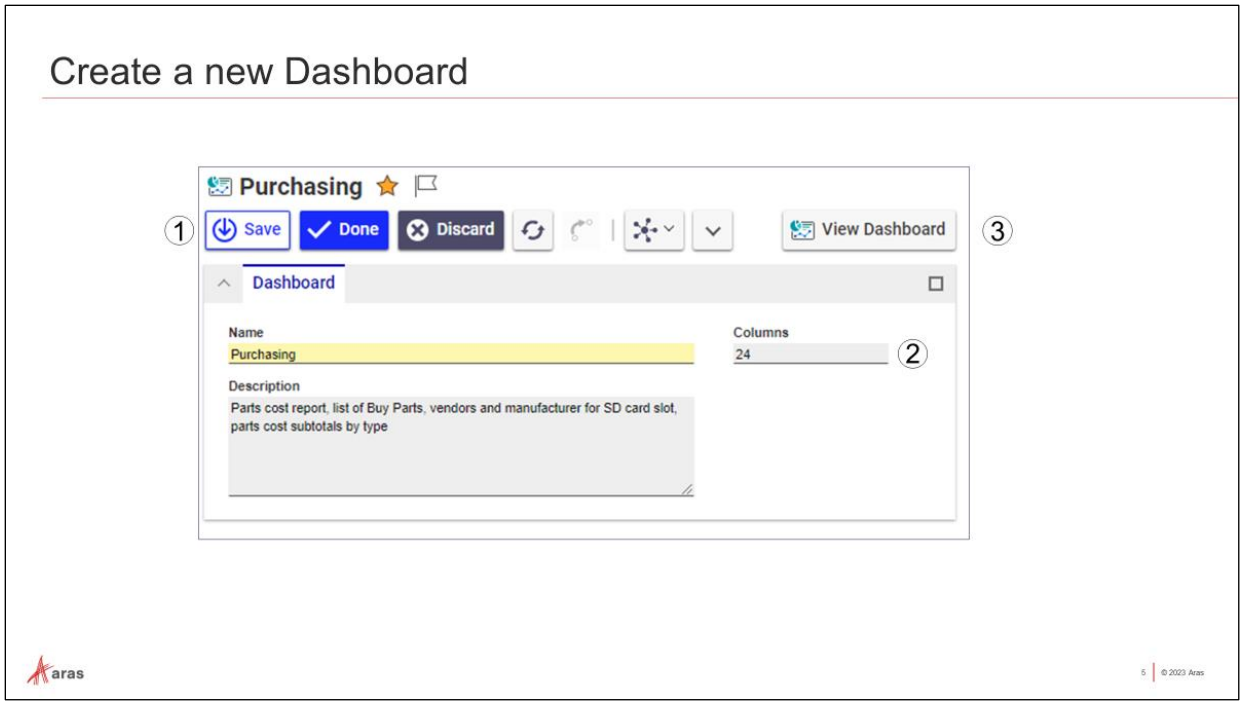
Dashboard Authors can view and create Dashboards. However, Dashboard Authors can only edit and delete Dashboards they create.

Dashboard Viewers can only view Dashboards, and only modify display settings (e.g., maximize a widget, browse pages, or search results), not modify the widget settings.

Users without one of these three identities do not have access to Dashboards, unless the permissions are customized, or a specific Dashboard has a private permission that overrides the default permission model.

Try it … Explore the cui_Dashboard permissions

1. Logged as your personal user, open the **cui_Dashboard** ItemType.
2. Analyze the content of ‘Can Add’ tab.
3. Under Permissions tab, open the **cui_Dashboard** default Permission.
4. Close the **cui_Dashboard** ItemType and related **cui_Dashboard** Permission.



Create a new Dashboard

While creating a new Dashboard, creators need to keep the number of columns to 24. This is used to define the total width of the dashboard. The dashboard content is responsive, i.e., the widgets will adapt in width depending on the browser size.

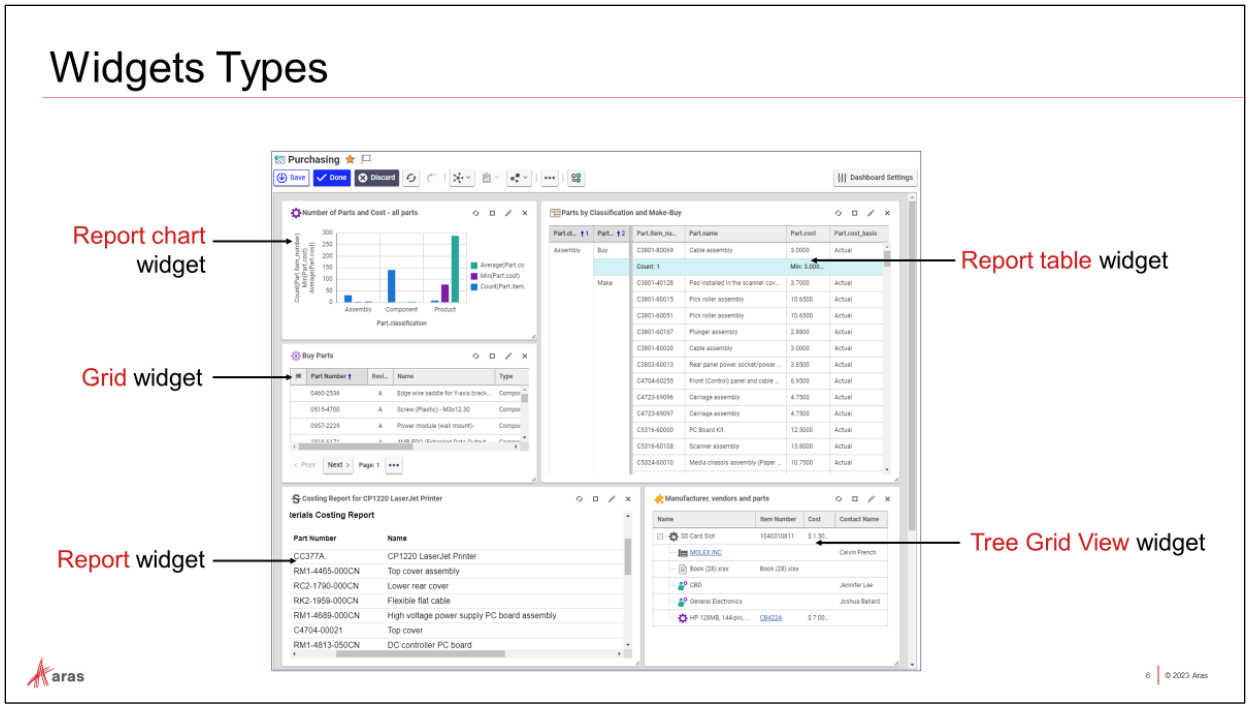
Try it … Create a new Dashboard

Throughout the try-its in this unit, we will create one Dashboard that we will fill with one widget of each type. This Dashboard is intended for the Purchasing community and gives some data visuals on part cost, buy parts and a tree grid view of a specific manufacturer part of interest.

1. Navigate to My Innovator > Dashboards, then create a new Dashboard.
2. Enter the following:

Name	Purchasing
Description	Parts cost report, list of Buy Parts, vendors and manufacturer for SD card slot, parts cost subtotals by type
Columns	24

3. Click Save.
4. A new button ‘View Dashboard’ appears on the top right-hand corner, click on it. The Dashboard canvas appears.
5. Notice that the top right-hand corner button is now ‘Dashboard Settings’.

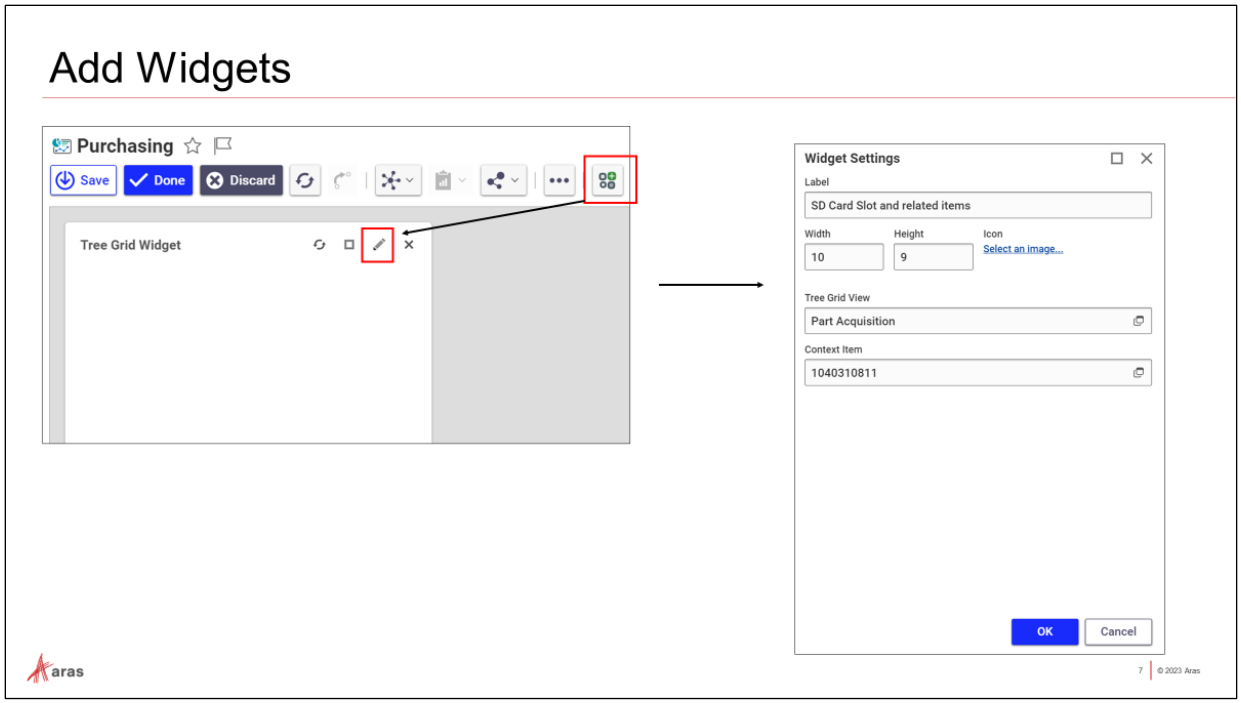


Widget Types

Dashboard Authors and Dashboard Administrators can add widgets to a Dashboard, using different widget types. They will first choose a widget template; each widget template defines the properties to be filled by the creator of a new widget. Remark: the training environment contains one widget template for each widget type, but having several widget templates for one widget type is supported.

Widget Type	Purpose	Requires a query definition	Requires a rpt_Report	Points to items of Item type
Grid	Show the results of a saved search	No	No	Targeted item type ; saved searches in Favorites item type
Tree Grid View	Show a Tree Grid View in context of an Item	Yes	No	Tree Grid View (rb_TreeGridViewDefinition)
Report	Display existing reports	Yes	No	Report
Report table	Show query definition results including related items data, possibly with totals/subtotals	Yes	Yes	rpt_Report
Report graph	Graphs of type column/bar charts and pie/donut charts, showing one or several measures on each property	Yes	Yes	rpt_Report

Note: do not confuse **Report** with **rpt_Report** item types: **Report** item type is used for SSRS reports, whereas **rpt_Report** item type is used for Report tables and Report charts from the Aras Reporting application in Dashboards.



Add Widgets

A new dashboard presents an empty canvas, where the creator (Dashboard Author or Dashboard Administrator) can add several widgets.

To do so, the creator will first select the widget type, then will choose a widget template. Then the creator will fill the widget settings. Those settings are pre-defined and depend on the widget template chosen in the previous step. Most of the settings are required for the widget to function properly.

Notes:

- on Grid widgets, the Search setting is optional: if empty, it searches all items of the selected item type.
- Grid Widgets in a Dashboard restore the Grid Layout saved with the selected Favorite Search, so that only the specific desired columns are shown.

Try it … Add a Tree Grid widget

1. Click on the Add Widget button.
2. In the search grid, select Widget Type=Tree Grid, search and select the corresponding widget template “Tree Grid Template”. Click OK.
3. A Tree Grid widget appears; click on its Edit button (pencil icon). Enter the following:

Label	Steel Flat-Head Socket Cap and related items
Width	10
Height	9
Icon	Select an image of your choice
Tree Grid View	Part Acquisition
Context Item	93395A213

4. Click OK.

Try it … Add a Report Widget

1. Click on the Add Widget button.
2. In the search grid, select Widget Type=Report, search and select the 'Report Template' widget template. Click OK.
3. On the new Report widget, click on the Edit Button.
4. Notice that the settings changed compared to the previous Tree Grid widget.
5. Enter the following:

Label	Costing Report for Makerbot Replicator
Width	14
Height	9
Icon	Select an image of your choice
Report	BOM Costing Report
ItemType	Part
Item	MP0101

6. Click OK.
7. On the Dashboard, click Save.

Try it … Create and share saved search and use it on new widgets

1. Navigate to My Innovator > My InBasket.
2. Click on Add to Favorites button, and enter the following:

Label	My Tasks
Share with	All Employees
Pin To Quick Access	Checked
Include Current Layout	Checked

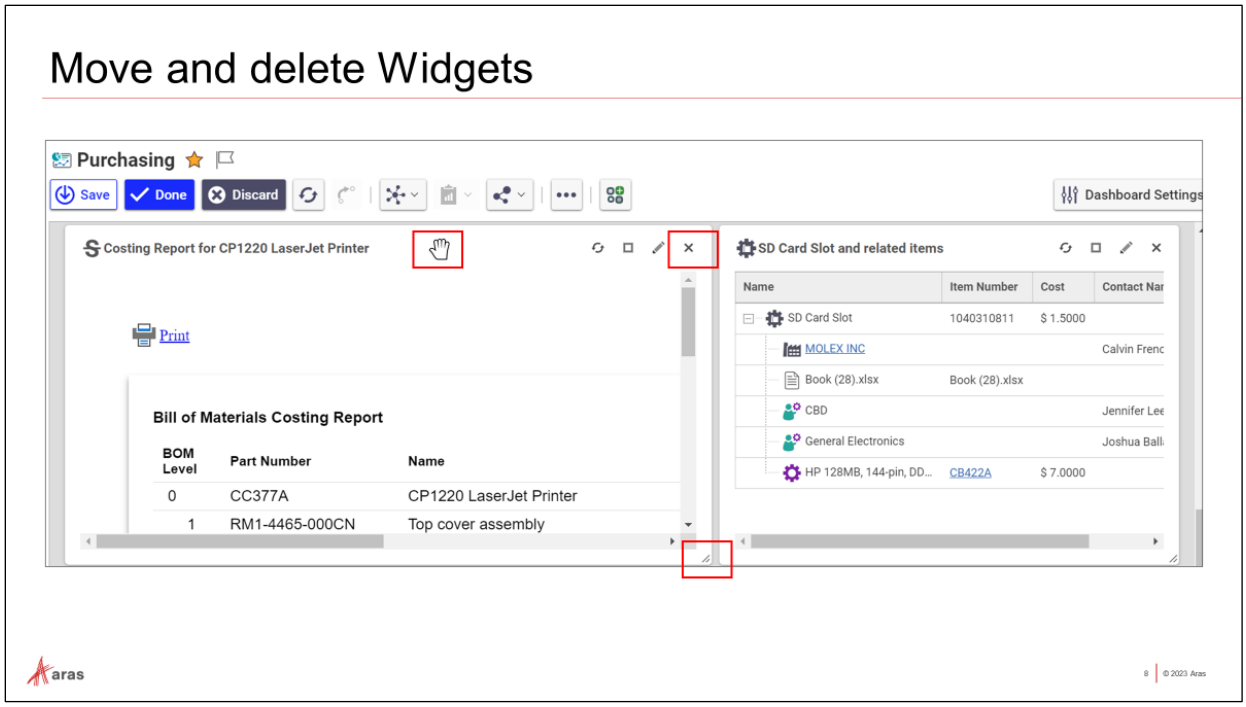
3. Click Save. Return to the Purchasing dashboard.
4. Click on the Add Widget button.
5. In the search grid, select Widget Type=Grid, search and select the 'Grid Template' widget template. Click OK.
6. On the new Grid widget, click on the Edit Button. Enter the following:

Label	My tasks
Width	10
Height	9
Icon	Select an image of your choice
ItemType	InBasket Task
Search	My tasks
Results per page	20
Maximum results	*empty*

7. Click OK.
8. Repeat the steps above to create a second grid widget, as follows:

Label	Buy Parts
Width	11
Height	7
Icon	Select an image of your choice
ItemType	Part
Search	Buy Parts
Results per page	20
Maximum results	*empty*

9. Click OK.
10. On the Dashboard, click Save.



Move and delete Widgets

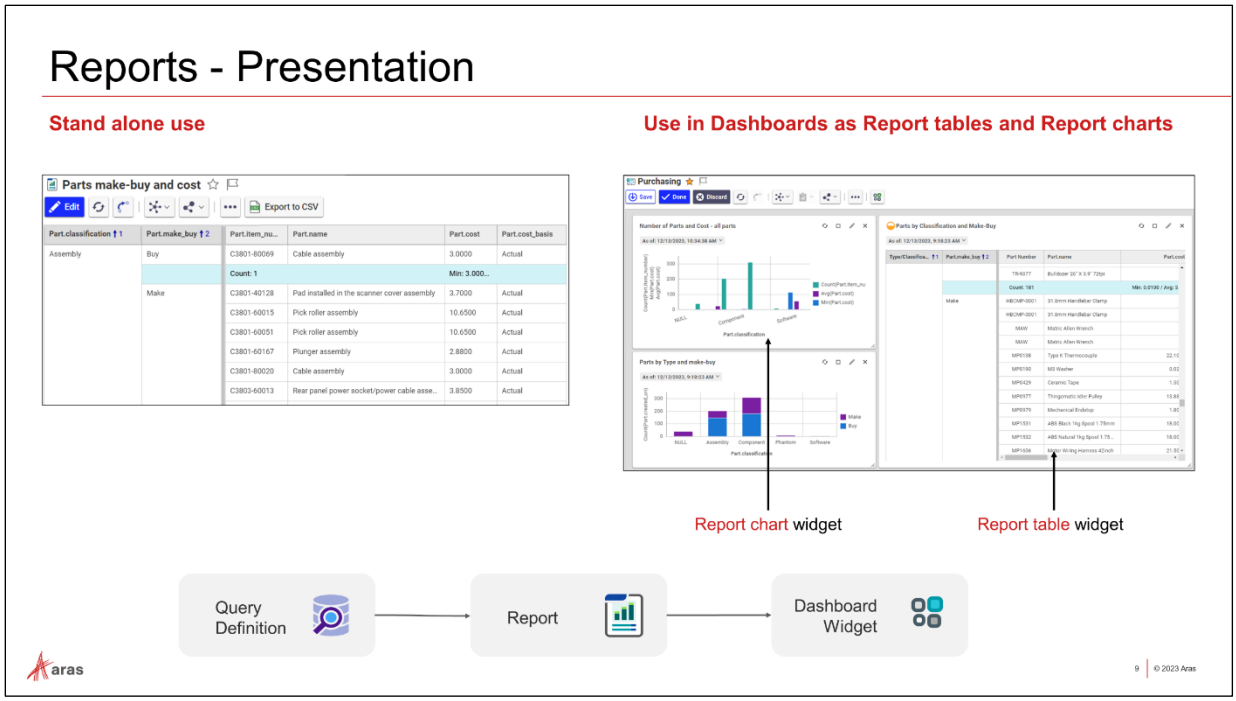
Widgets can be dragged and dropped in other locations of the Dashboard canvas, by hovering over the widget header (the mouse turns into a hand).

The widget width and height can also be changed directly on the dashboard canvas by clicking on the bottom-right corner of the targeted widget.

Finally, clicking on the cross on the top-right corner of a widget deletes it.

Try it … Change the widgets localization on the Dashboard

1. On the **Purchasing** Dashboard still in editing mode, put the report widget and the tree grid view widget side-by-side.
2. Place the “My tasks” grid widget at the bottom left.
3. Click Save.



Reports - Presentation

Reports are formatted and interactive, intended to show Aras Innovator data in different combinations, show it in a user-friendly format with the capability to share the result. Reports are based on a query definition, which allows the report to show data from context items as well as data from their related items.

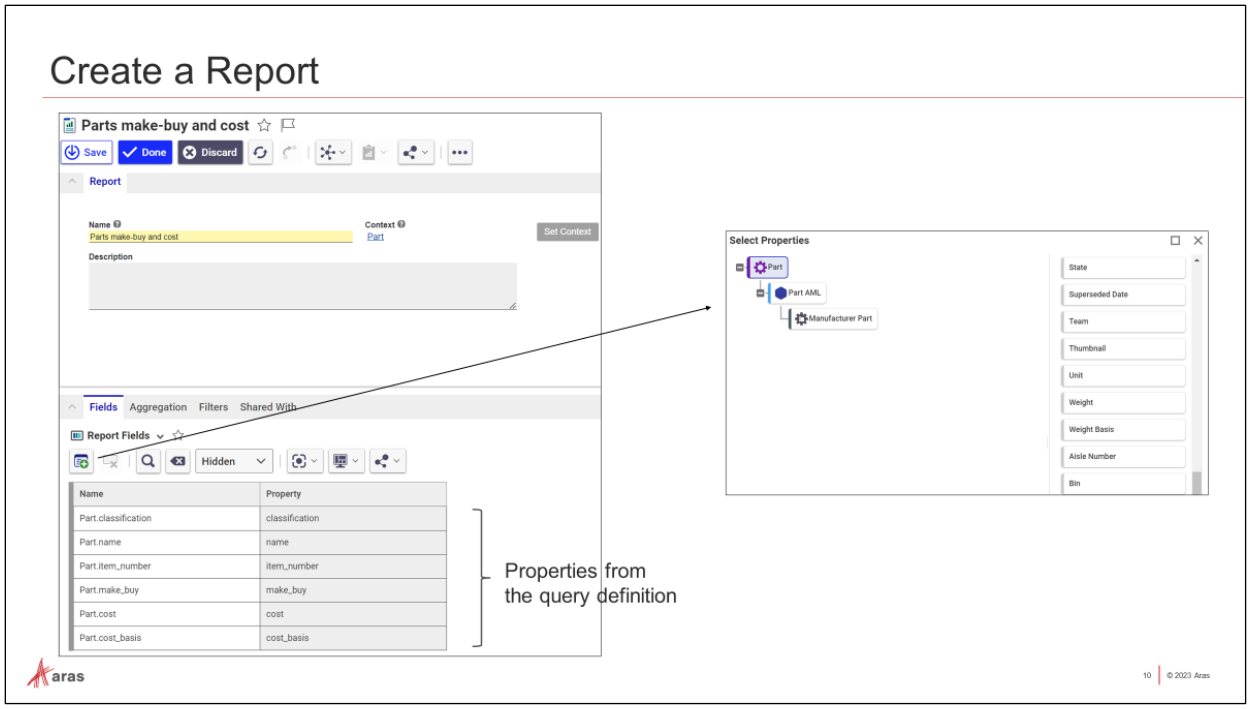
The new Report item type (rpt_Report) can be used for multiple purposes:

- Stand alone: users can see and export data as a table. The rpt_Report items allow to group rows of results according to one or more properties and show subtotals and totals.

- use within dashboard to build report tables: the same report is displayed in a dashboard, as a widget

- use within dashboard to build report charts: the report charts rely on data coming from a report. Then the widget creator can define the chart display.

This data model promotes re-usability: Dashboard creators can build several reports out of the same query definition and several dashboard report tables/charts out of the same report.



Create a Report

Only Report Authors group Identity can create Reports. In the training environment, the Administrators identity is part of that group identity.

The report is created based on a selected Query Definition. It will then use the relationships defined in that Query definition.

When creating a report, first select the Context Item Type. Then, select the Query Definition: that selection is restricted to the Query definitions configured using the same Context Item Type.

That selection imports the configuration of the Query Definition into the Report, such as:

- The query definition structure of item types and relationships
- The properties configured in the query definition, which appear in the Report's Fields tab. Report authors can add any properties and extended properties of all item types present in the Query Definition structure of related items, even if they were not configured in the query.

Upon save, the Report breaks the relationship to the original query definition. The Report becomes independent of any modification of the originally selected Query. This setup avoids query definition updates to break the Report and is a preparation for a future product functionality.

Try it … Create a Report

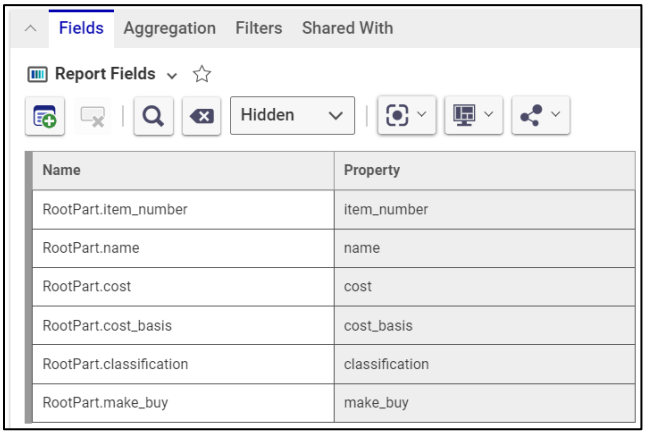
In this try-it, you will use your personal user who is part of the Administrators group identity, which is itself part of Report Author. You use a pre-existing query definition on Parts which you use to pull configured properties and add other Part properties. You will use this report to build the report tables and report charts later.

1. In the TOC, go to My Innovator > Reports.
2. Create a new Report, as follows:

Name	Parts make-buy and cost

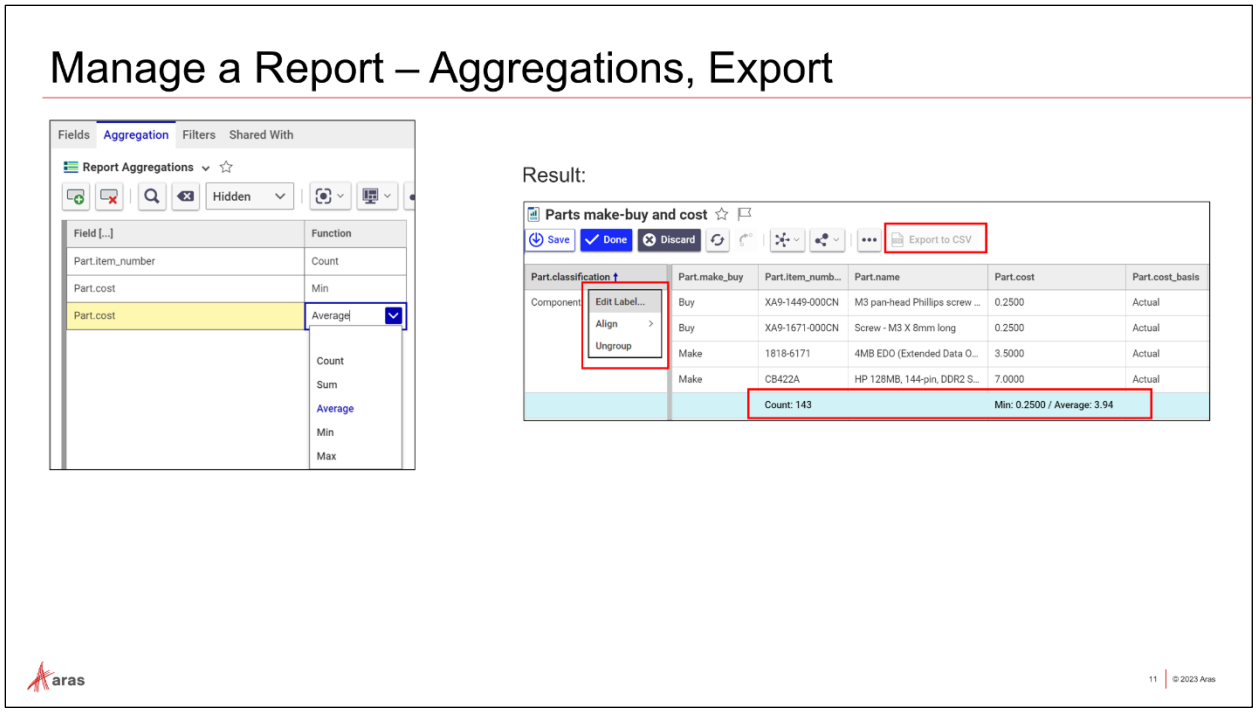
Description	Parts type, cost, make-buy - from PE_costingReport

3. Click Set Context.
4. Choose Design > Parts. Click OK.
5. Search and select the **PE_costingReport** query definition.
6. Click Save.
7. On the Fields tab, leave the four first fields (name starting with “RootPart.”) and delete the other fields.
8. Click Add Report Fields, select RootPart in the query structure.
9. Select both Type and Make-Buy properties, using the Ctrl key. Click OK.
10. Click Save. The result should look like this:



Name	Property
RootPart.item_number	item_number
RootPart.name	name
RootPart.cost	cost
RootPart.cost_basis	cost_basis
RootPart.classification	classification
RootPart.make_buy	make_buy

Note: do not modify the Name of the fields in this location. We will change some labels later on using the Preview.



Manage a Report – Aggregations, Export

From Aggregation tab, the Report Author can call calculations on fields: Count, Sum, Average, Min, Max:

- Count is available for Text/String/Date fields.
- Count, Sum, Average, Min, Max are available for numerical fields.

The same field can be used multiple times for several aggregations, e.g., a Decimal field Cost can have both Min, Max, and Average aggregations.

The aggregation(s) will appear on the Report Preview in the totals.

If the Report Preview has a group defined on one of several fields, then subtotals will be calculated on each subgroup. Groups are defined by right-clicking on the column header of a field. Several groups can be defined in one Report: in this case the order of grouping fields by the Report Author defines the hierarchy of groups.

The subtotals and totals allow users to analyze data trends, compare subtotals for each set against the overall total.

A Report Item can be used to visualize a dataset based on selected fields, configured filters, and aggregations/groups. The Preview shows the full resulting dataset and can be exported. The Preview layout can be modified in Edit mode: column order, column width, data alignment in cells and column label.

The export of the Report results is on CSV format: that format was chosen for dataset export as it is the most generic. It is accessible through the ‘Export to CSV’ button.

Try it … Define aggregations and groups on a Report

1. In the *Parts make-buy and cost* Report, go to Aggregations tab, click on New Report Aggregation button and fill as follows:

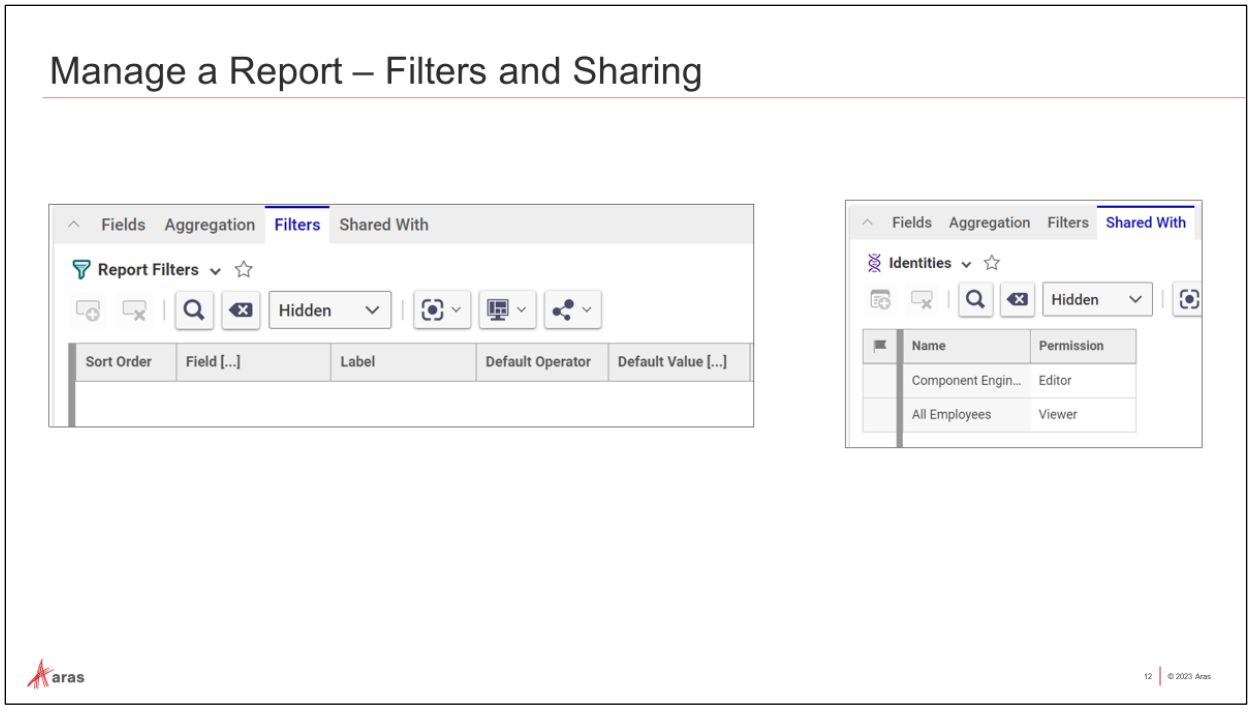
Field	Function
RootPart.item_number	Count
RootPart.cost	Min
RootPart.cost	Avg

2. Click Save.
3. Click the Preview button on the top-right corner.
4. Right-click on the following column headers and select Group for “RootPart.classification” then “RootPart.make_buy”.

   Notice that both columns move to the left-hand side and a freeze column appears on their right.
5. Right-click on the *RootPart.classification* column, select Edit Label and re-label it *Type*.
6. Right-click on the *RootPart.item_number* column, change its Label to *Part Number*.
7. Right-click on the *RootPart.cost* column, select Align > Center.
8. Scroll down and notice the subtotals and total, each displaying Count, Min, and Average.
9. Click Done.

Try it … Export the Report

1. Click on the ‘Export to CSV’ button.
2. Notice the raw format of the dataset in CSV: this allows further manipulation in Excel later (e.g., pivot tables).



Manage a Report – Filters and Sharing

Report Filter represents a filtering condition for Report Dataset based on Field values. Filtered out records are not included in the resulting DataSet.

Sort Order	Filters are applied in sequence order with the “AND“ logic operator
Field	Used as the first argument for the Filter’s Operator
Label	Add a custom label. By default, the Field value is used.
Default Operator	The data type of the added Report Field dictates the list of available operators
Default Value	Used as the second argument for the Field’s Operator

The Share With tab allows the Report Author to grant editing and Viewing access to group identities:

- Editor: corresponds to the *Full Access Sharing*. By assigning this role, the identities will be able to execute, update and delete the Report.
- Viewer: corresponds to the *Read Only Sharing*. By assigning this role, the identities will be only able to discover and execute the Report.

Try it … Manage Filter and Sharing with on a Report

1. Click on Report Settings in the top-right corner, then click Edit.
2. On the Filters tab, click New Report Filter and fill as follows:

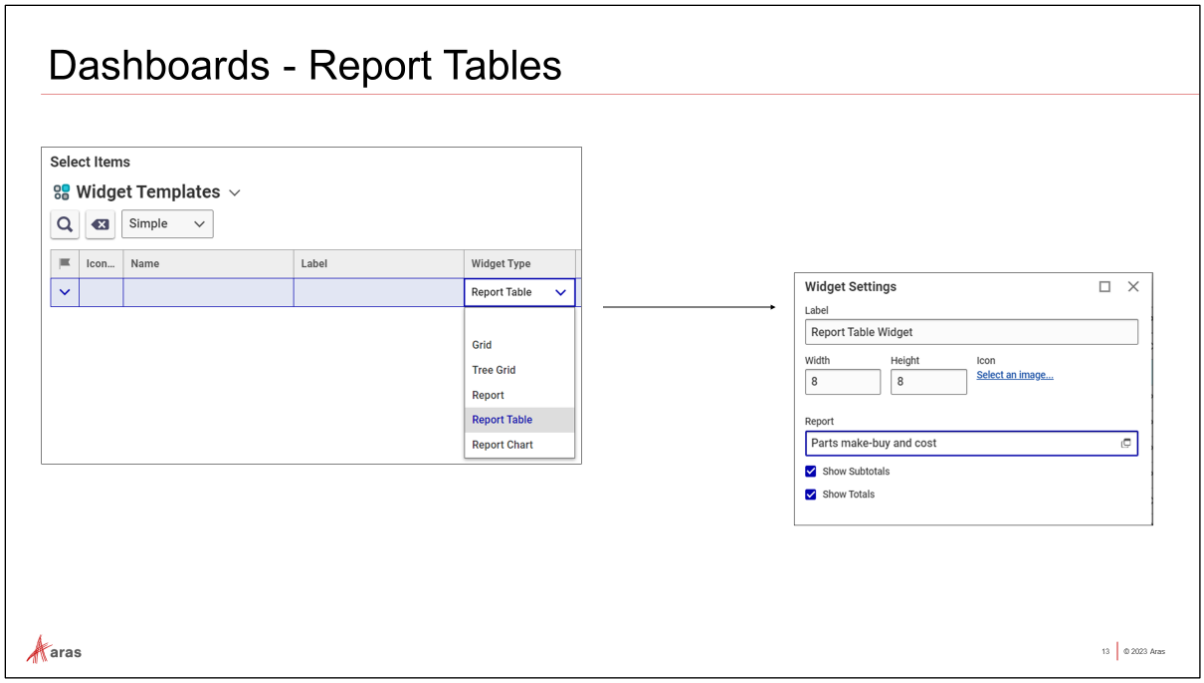
Sort Order	Field	Label	Default Operator	Default Value
10	RootPart.cost	RootPart.cost	>=	5

3. Click Save.
4. Go to Preview and observe the result.
5. Go back to Report Settings and delete the filter.

6. Go to the *Shared With* tab, clear the search criteria, then fill as follows:

Name	**Role**
Administrators	Editor
Component Engineering	Viewer

7. Click Done.



Dashboards – Report Tables

Reports can also be used to display data using a widget in a Dashboard. One way is to select the Widget type=Report table, which will display the Report preview in the resulting widget.

Note that users will need the Viewer or Editor access to the report item, in addition to the access to the dashboard itself, to see the report table in the dashboard.

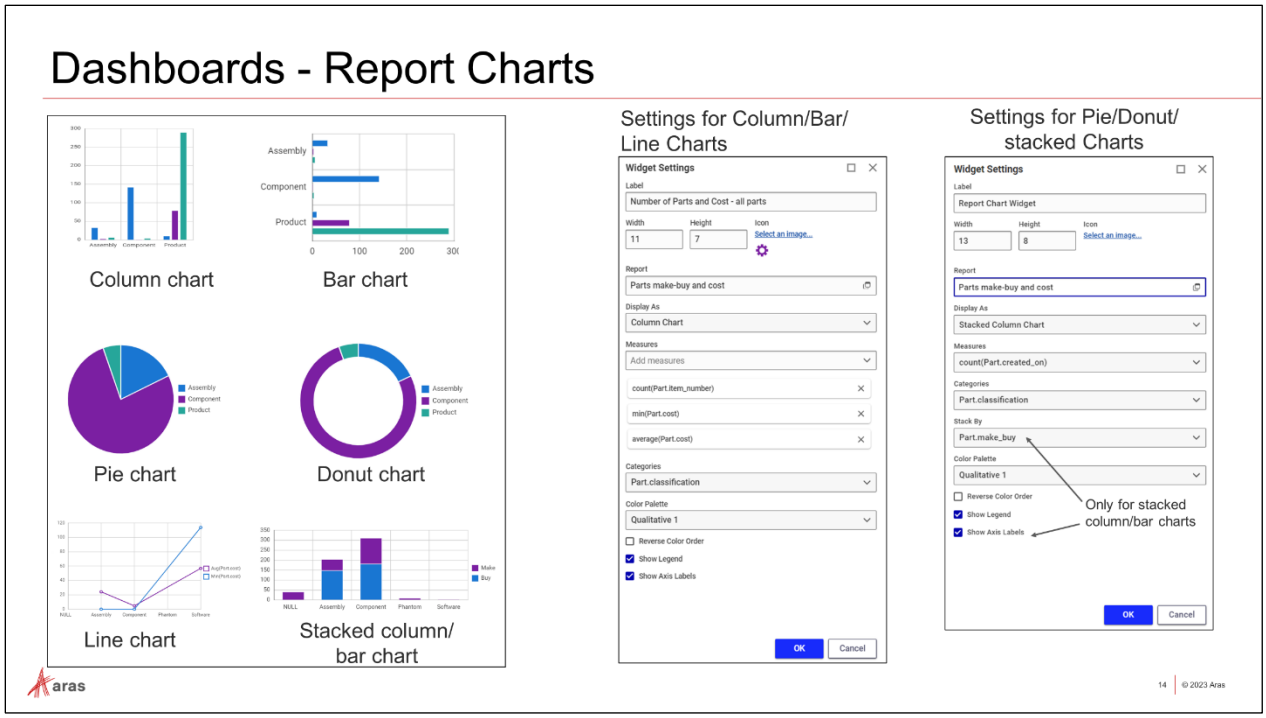
Upon widget creation, the creator will select the report item and choose if the report table widget will display the subtotals and/or totals, in addition to the usual widget settings (label, width, height, icon).

Try it … Create a report table in a Dashboard

1. Open the **Purchasing** Dashboard for editing.
2. Click ‘Add widget’, select Widget Type=Report Table, select the ‘Report Table Template’ widget template and click OK.
3. On the new widget, click on the Edit button.
4. Fill as follows:

Label	Parts by Classification and Make-Buy
Width	13
Length	14
Icon	Select an image of your choice
Report	Parts make-buy and cost
Show subtotals	Checked
Show totals	checked

5. Click OK. Notice that a message appears the first time we display a report with a widget, because the rpt_report data needs to be generated for the user.
6. Click the 'Generate Current Results' button. The subtotals, totals, and groups appear like in the Report Preview.



Dashboards – Report Charts

Alternatively, the Report items can also be used to create widgets of type Report charts. This allows us to build graph charts, such as column or bar charts, and pie or donut charts. This grants users with tools for data visualization and decision-making. Note that one Report item can be used in several widgets to display data in different ways.

For Pie and donut charts, the creator will select one measure coming from a field aggregation of the related report.

For column charts, bar charts and line charts, the creator can select **several** measures shown side-by-side. The axis label can be hidden or shown.

On all charts, data can be split by categories. Furthermore, several the color palettes are available natively, the creator can configure the chosen one and reverse color order. The creator can also choose to show or hide the legend.

Try it … Create a Column Chart

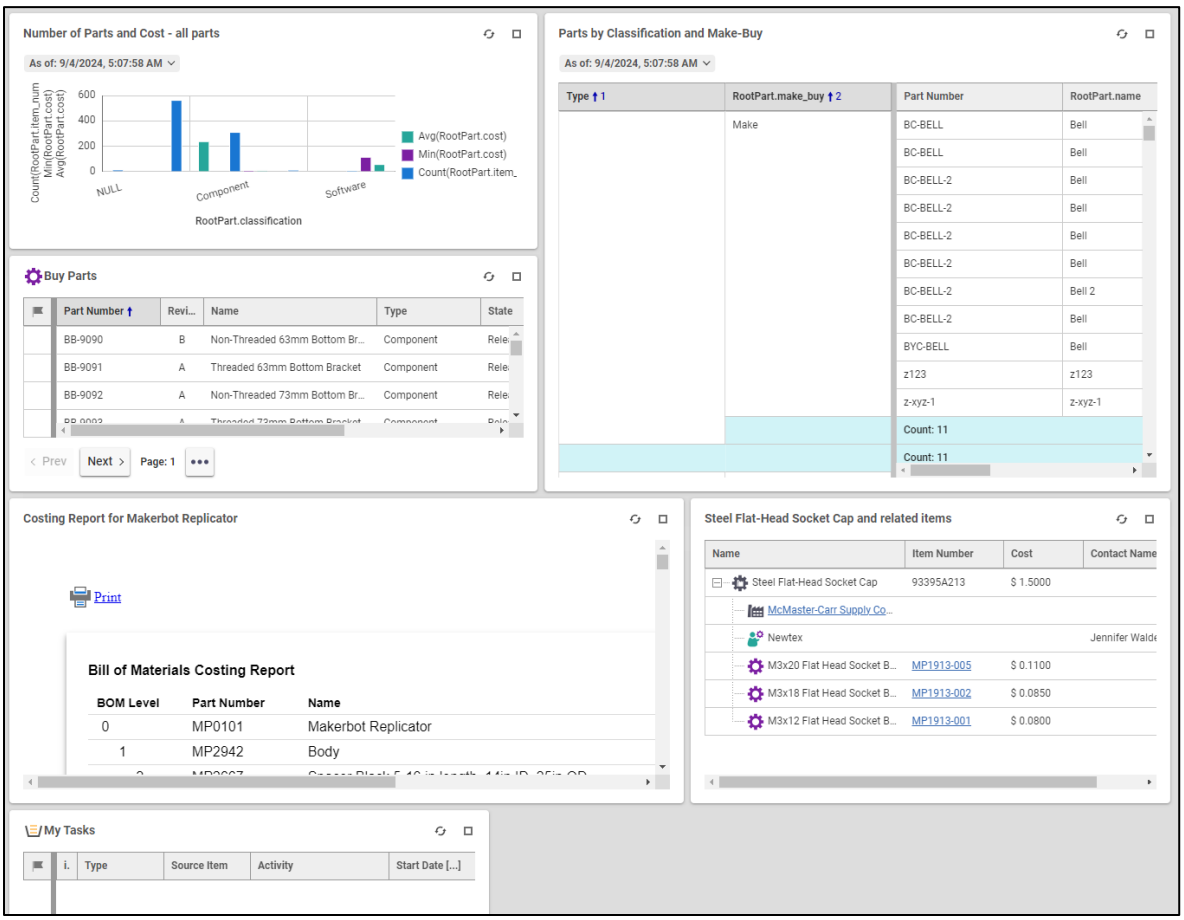
In this exercise, you use the run a wizard to create a new Relationship tab for our TGV.

1. Click on the Add Widget button.
2. In the search grid, select Widget Type=Report Chart, search and select the 'Report Chart Template' widget template. Click OK.
3. On the new widget, click on the Edit Button.
4. Enter the following:

Label	Number of Parts and Cost - all parts
Width	11
Height	7
Icon	Select an image of your choice

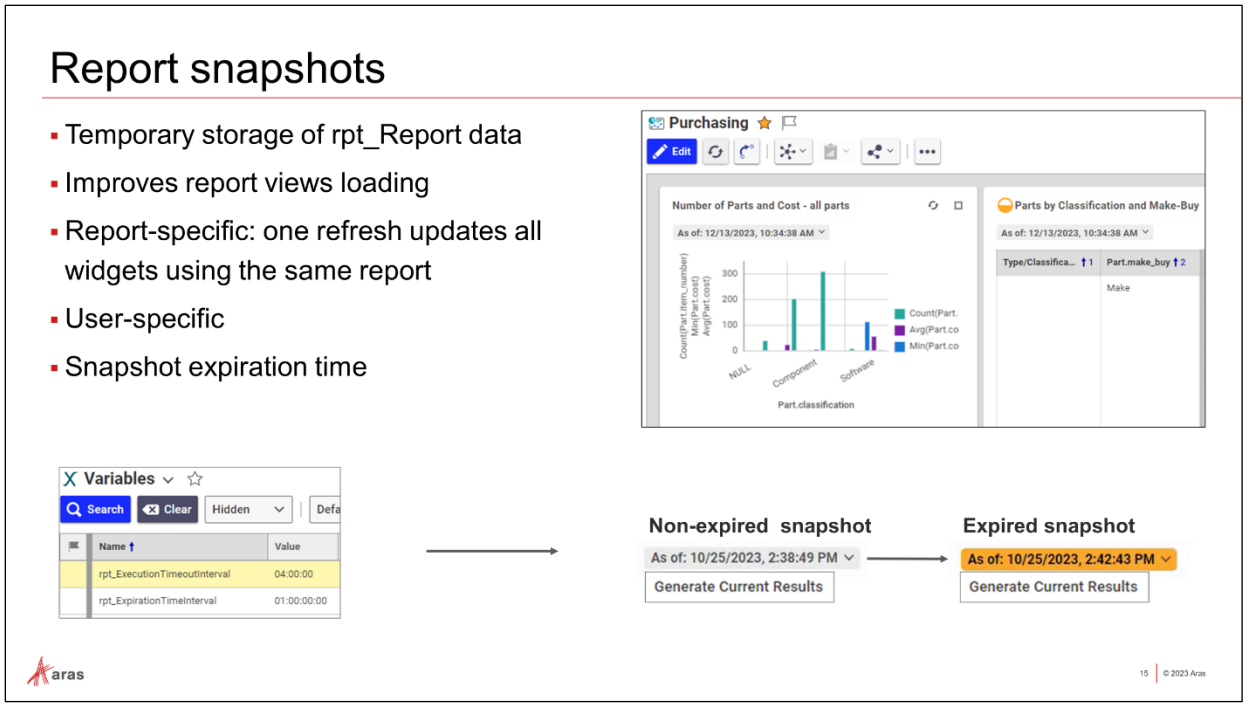
Report	Parts make-buy and cost
Display as	Column Chart
Measures	Count(RootPart.item_number) min(RootPart.cost) average(RootPart.cost)
Categories	RootPart.classification
Color Palette	Qualitative 1
Reverse Color Order	Unchecked
Show Legend	Checked
Show Axis Label	Checked

5. Click OK.
6. Reorganize the widgets to improve the display. The result should be as follows:



7. On the Dashboard, click Save.

   Note: at the moment, the dashboard is visible by dashboard administrators, dashboard authors and dashboard viewers. You can optionally create a private permission to modify this setup.



Report snapshots

Reporting introduces a new mechanism to temporarily store the Report execution by User. This improves the loading of report views in the dashboard environment. Users can at any time identify the execution time of the data they are reviewing, under the Widget label. They can manually request a new execution to retrieve the latest data, by clicking on *Generate Current Results*. This applies to widget types created from rpt_Reports, i.e. report charts and report tables ; it does not impact the following widget types: Grid, Tree Grid View, and report.

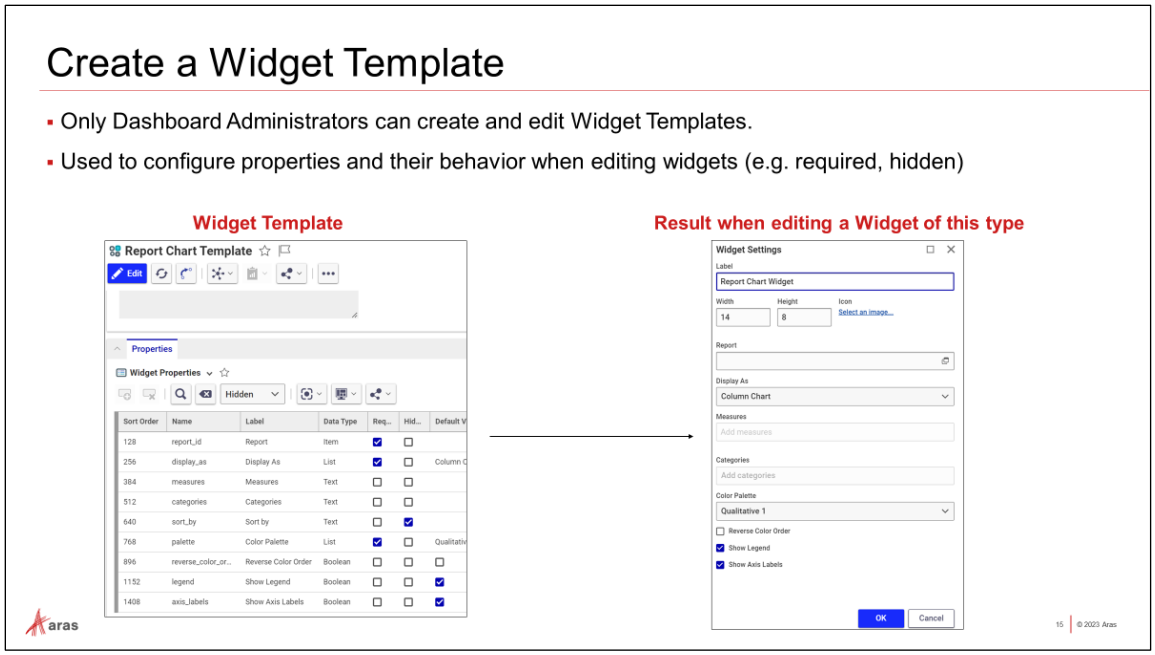
The snapshots are report-specific, which impacts all dashboard widgets using the same report. They are also user specific.

The benefits of the implementation of reporting snapshots include:

- Rendering faster reports by reporting on data that is already summarized.
- Creating dashboards that refresh quicker.
- Synchronization of Report executions to avoid multiple execution requests for the same report.

An administrator can configure the snapshots to have a timeout and an expiration time:

Variable	Variable name	Purpose
Timeout	rpt_ExecutionTimeoutInterval *(expressed as hh:mm:ss)*	the Report execution exits from a Report Execution if not getting the results in a reasonable time
Expiration time	rpt_ExpirationTimeInterval *(expressed as dd:hh:mm:ss)*	defines the time after which a stored Report snapshot is labeled as expired (shown in orange on Dashboard widgets), requiring users to Generate New Results



Create a Widget Template

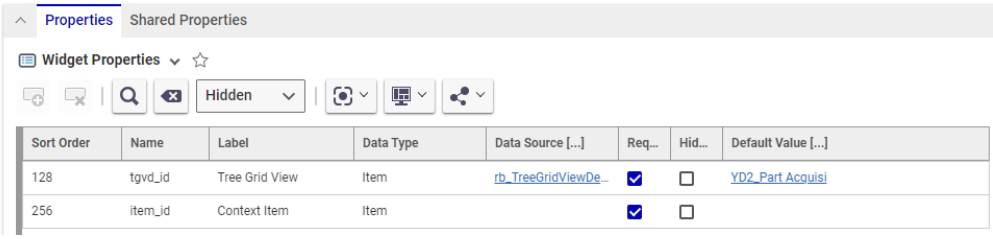
Widget templates define the properties that will be displayed when creating a widget of this type, along with the property's behavior (required, hidden, etc.).

Natively, one widget template exists for each widget type. But the Dashboard Administrators can create new widget templates of the same type, to help the dashboard authors and administrators to create new widgets quicker. Indeed, the additional templates can offer pre-configured widgets to the widget creators down the road.

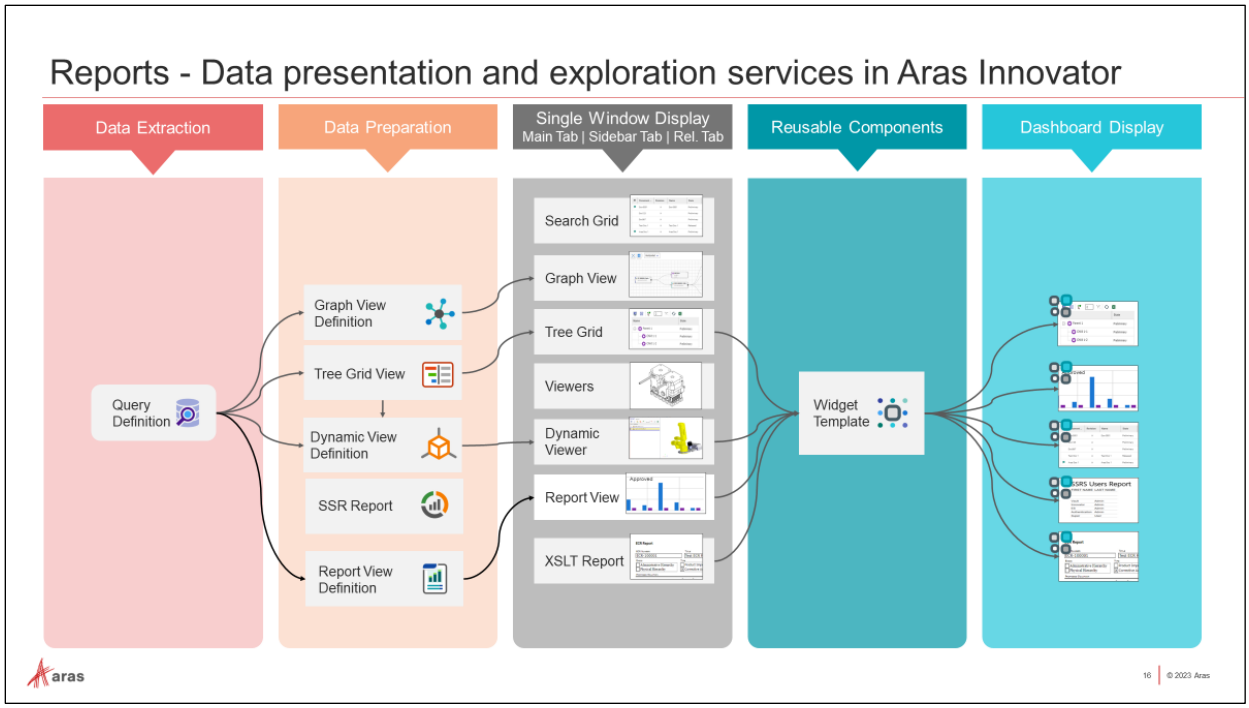
For this, the additional widget templates can populate pre-configured values using "default value" and hidden fields, therefore restricting the choice of widget creators for specific purposes.

Try it … Create a Widget Template

1. Go to Administration > Configuration > Widget Templates.
2. Open a search grid for Widget Templates and run a search.
3. Right-click on the Tree Grid Template, then More > Save as.
4. Click Edit, then change the Name to **Tree Grid - Manufacturers and relationships template**.
5. Go to the properties tab. On the first row 'Tree Grid View', change the Default Value to **Part Acquisition**. Click Done and unclaim the widget template.



6. On the **Purchasing** dashboard, create a new widget of type Tree Grid, select the template **Tree Grid - Manufacturers and relationships template.** Notice that the Tree Grid View item is already selected for the widget creator. The creator only needs to populate the Context Item.
7. Click Cancel. On the widget window, select Delete. On the Dashboard, click Done.



Reports Data presentation and exploration services in Aras Innovator

This is a summary of the objects created out of a query definition on one side, and the different types of widgets on the other side.

The different item types allow re-usability: for example, we can create several reports out of one query definition.

Summary

In this unit, you learned how to create a Query Definition that populates a Tree Grid View control.

You should now be able to:

- Manage Dashboards and widgets
- Understand the Dashboard security access
- Create, move, and delete widgets
- Manage Reports
- Export a report preview
- Create Report tables and report charts
- Manage a widget template

Review Questions

1. Who can create dashboards ?
   - Report Authors
   - Dashboard Authors
   - Dashboard Viewers
   - Dashboard Administrators

2. A report can be used:
   - on a tree grid view for a context item
   - on its own, to display data with Preview
   - to create a report table widget in a dashboard
   - to create a report chart widget in a dashboard
   - to define search criteria on a grid widget
   - all of the above

3. What is the purpose of creating additional widget templates ?

Lab Exercise

Goal

Create a Report item that imports the structure of an existing Query Definition, add fields; create a dashboard with widgets of each widget type.

Scenario

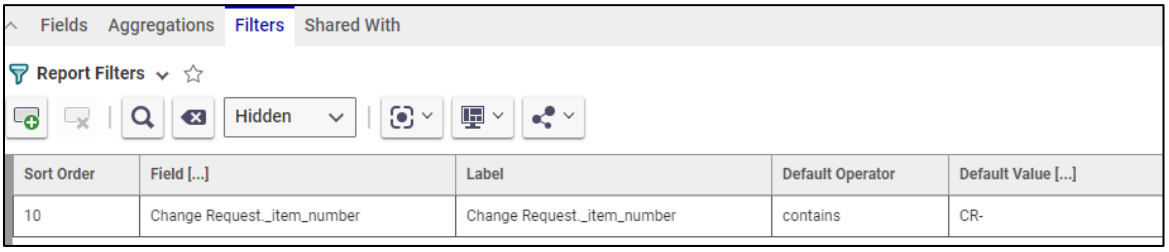
In this exercise, you will use the Query Definition Customer Requests to build a Report and add the Region, Design Centers, and State in the report fields. The Dashboard will be used by Change Request Managers, part of the Component Engineering group, to show Change Requests data.

Steps

Ensure you are logged into Aras Innovator as your personal user.

Create a Report

1. Create a new Report named *Change Requests with Customers*, choose **Customer** as the Context ItemType, select the Query definition *Customer Requests*, and save the changes.
2. In Fields tab, click Add Report Fields, then fully expand the structure to add additional fields by selecting these Change Request properties: **Design Center** and **Region**.
3. In the Aggregations tab, add the field **Change Request._item_number** with function=Count.
4. In the Filters tab, add the field **Change Request._item_number** with Default Operator=contains and Default Value= ”CR-“.



5. In Shared With tab, add the following:

Name	Permission
Administrators	Editor
Engineering	Viewer

6. Click Save.
7. Go to Preview, then right-click on the **Region** column header and select *Group*. Repeat this step for **Design center** field.
8. Click Done.

Create a saved search

9. Go to Product Management > Improvement > Change Requests.
10. Create a saved search named **Latest Change Requests**, shared with All Employees.

Create a Dashboard and add widgets of different types

11. Create a new Dashboard named **Change Requests** and with description= For Component Engineering. Keep the columns to 24. Click Save.
12. Switch to View Dashboard, then create a widget of type=Report, as follows:

Label	Quantity Rollup Report for MakerBot replicator with cover
Width	8
Height	24
Icon	Select an image of your choice
Report	BOM Quantity Rollup Report
ItemType	Part
Item	MP0103

13. Create a widget of type=Tree Grid, as follows:

Label	CR's from Delphi Corporation customer
Width	8
Height	8
Icon	Select an image of your choice
Tree Grid View	Customer Requests
Context Item	Delphi Corporation *(you are free to select another Customer Item instead, and pick the Customer you related most frequently to your CR's during test)*

14. Create a widget of type= Grid, as follows:

Label	List of all CR's
Width	8
Height	8
Icon	Select an image of your choice
ItemType	Change Request
Search	Latest Change Requests
Results per Page	20
Maximum Results	empty

15. Create a widget of type= Report Table, as follows:

Label	CR's by Region and Design Center
Width	16
Height	8
Icon	Select an image of your choice
Report	Change Requests with Customers
Show subtotals	unchecked
Show totals	checked

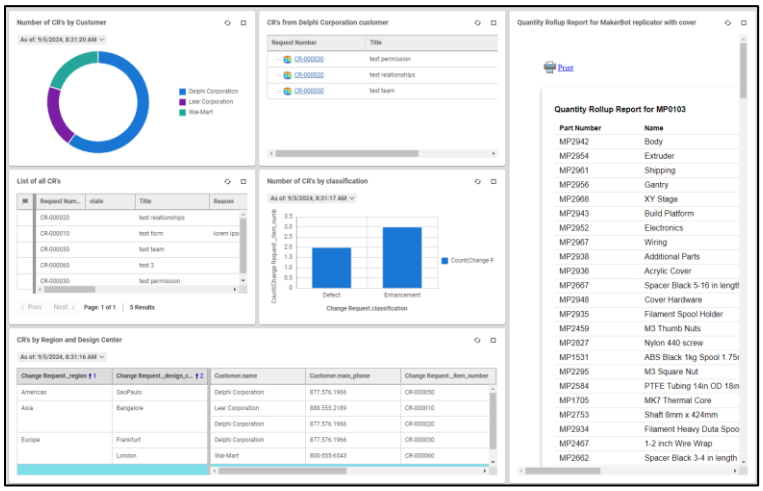
16. Create a widget of type= Report Chart, as follows:

Label	Number of CR's by classification
Width	8
Height	8
Icon	Select an image of your choice
Report	Change Requests with Customers
Display as	Column Chart
Measures	count(Change Request._item_number)
Categories	Change Request.classification
Color Palette	Qualitative 1
Reverse Color Order	Unchecked
Show Legend	Checked
Show Axis Label	Checked

17. Create a second widget of type= Report Chart, as follows:

Label	Number of CR's by Customer
Width	8
Height	8
Icon	Select an image of your choice
Report	Change Requests with Customers
Display as	Donut Chart
Measures	count(Change Request._item_number)
Categories	Customer.name
Color Palette	Qualitative 1
Reverse Color Order	Unchecked
Show Legend	Checked

18. Reorganize the widgets ; the result should look as per example below. Click Done.



This page intentionally left blank

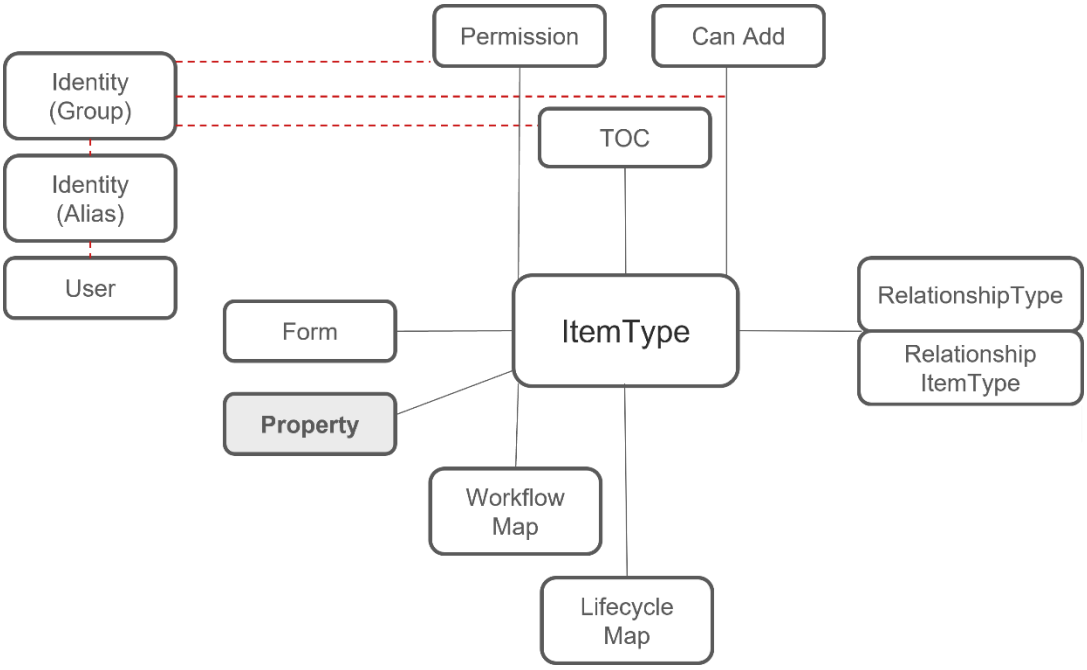
Unit 12 Defining Extended Classification

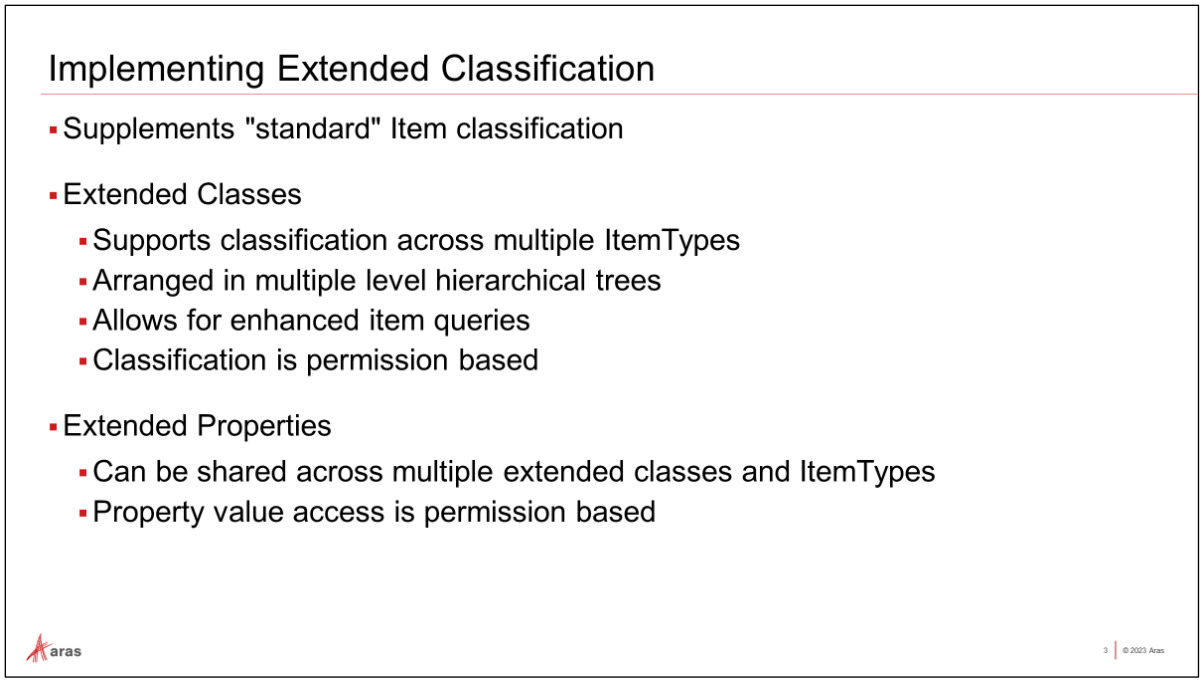
Overview

In this unit, you learn how to create and assign Extended Properties to one or more ItemTypes using Extended Classes. Extended Properties remove some of the restrictions imposed by standard Properties and allow for greater control over security.

Objectives

- Define Extended Properties and Classes
- Explain the responsibilities of the Classification Administrator
- Create Extended Properties and Classes
- Create and Manage Extended Classification Trees
- Set an xClass/xProperty private Permission





Implementing Extended Classification

Extended Classification (xClassification) is a supplemental feature to standard classification. Remember that standard classification allows you to further describe what kind of item is being created and to assign class path properties that align with that type of item (e.g., Part/Component or Part/Assembly). Standard classification only works within a single ItemType, and standard Properties cannot be shared across multiple ItemTypes.

However, Standard classification still has its own benefits not available through xClassification, like different Lifecycle States, Permissions, History Templates, and so on.

Extended Classification compliments standard classification by allowing you to create Extended Classes that can be arranged in hierarchical trees and assigned to multiple ItemTypes. Extended Properties can then be assigned to one or more Extended Classes, which allows you to share a property definition across multiple ItemTypes as well.

Extended Classification also offers more security features including the ability to configure who can classify, unclassify, or even see classifications of an item, as well as support for individual property level security.

In this unit, you learn how to create and maintain Extended Classes and Extended Properties, and you classify items using these classes and properties.

Describing Extended Property Types

- Extended Properties (xProperties) are defined on a global scope and are not specific to any one ItemType.
- Implicit
  - xProperty is assigned to xClass hierarchies that are configured for one or more ItemTypes
  - Users provide values using xClass Form control
- Explicit
  - xProperty is assigned to one or more ItemTypes directly
  - No standard UI provided (requires programming)

aras

4 | © 2023 Aras

Describing Extended Properties

Extended Properties supplement the standard Properties we discussed earlier in the course. Unlike standard Properties, Extended Properties (referred to as xProperties) are created in a global pool and can be assigned to multiple Extended Classes (referred to as xClasses) in a hierarchical tree. Extended Classification Trees can be created which are assigned to one or more ItemTypes.

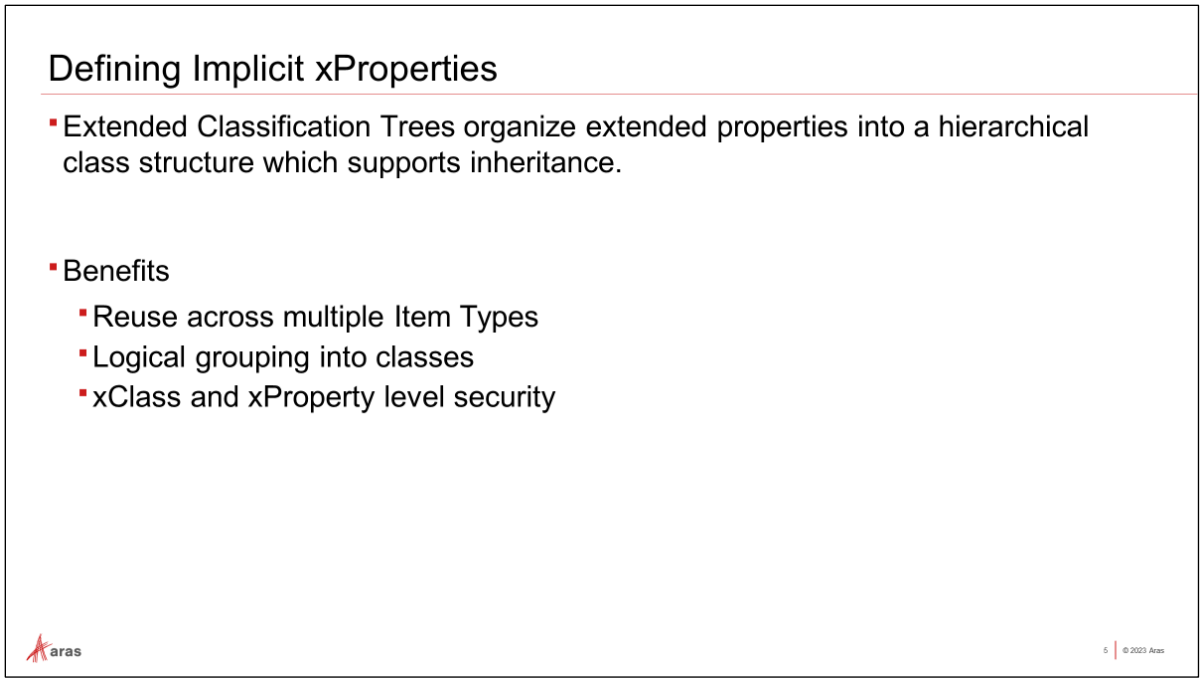
This flexibility removes many of the constraints of standard classification classes and properties and provides additional features for securing and finding data.

Implicit xProperties

Implicit xProperties are assigned to xClasses in an xClassification tree. xClassification trees are then assigned to one or more ItemTypes. You learn how to create and manage implicit xProperties in this unit.

Explicit xProperties

Explicit xProperties are assigned directly to one or more ItemTypes. Explicit xProperties are designed to be used by programmers familiar with the Aras AML language and are outside the scope of this course.



Defining Implicit xProperties

Implicit xProperties give a Classification Administrator the flexibility to assign groups of common characteristics/properties to Items of different ItemTypes. Unlike standard ItemType properties, Extended Properties can be shared across many Item Types and grouped into logical collections. An xProperty is defined in a global pool and is then associated to one more xClasses in an xClassification Tree.

xClassification Trees reference and organize the Extended Properties into a hierarchical class structure which allows for xProperty inheritance.

Extended Properties and Extended Classification are beneficial to:

- Manage properties across multiple ItemTypes
- Group properties into logical collections (xClasses)
- Provide property level and xClassification tree level security using xProperty permissions.

Comparing Classification Terms

Standard Classification		Extended Classification	
Term	**Definition**	**Term**	**Definition**
Property	Defined within a single ItemType (cannot be shared) and can be assigned to a Class	xProperty	Defined on a global scope and are not specific to any one ItemType
Class	Defines a category within a single ItemType and can be associated with properties in that single ItemType only	xClass	Defines a category of shared xProperties arranged in a hierarchical xClassification Tree
Class Structure	Hierarchical collection of Classes consisting of one Root level ItemType Class, and subclasses. A Class Structure is specific to one ItemType.	xClassification Tree	Hierarchical collection of xClasses consisting of one Root xClass, and Branch and Leaf xClasses. xClassification Trees are also defined on a global scope and are not specific to any one ItemType
Class Path	Hierarchy to indicate where a Property resides in a Class Structure	xClass Path	Hierarchy to indicate where an xProperty resides in an xClassification Tree

aras

6 | © 2023 Aras

Comparing Classification Terms

To distinguish between the terms used with standard classification, a prefix of "**x**" is prepended to the terms relevant to extended classification.

xProperty

Defines data to be stored in the database. Implicit xProperties are not associated directly with a single ItemType but reside in a global pool.

xClass

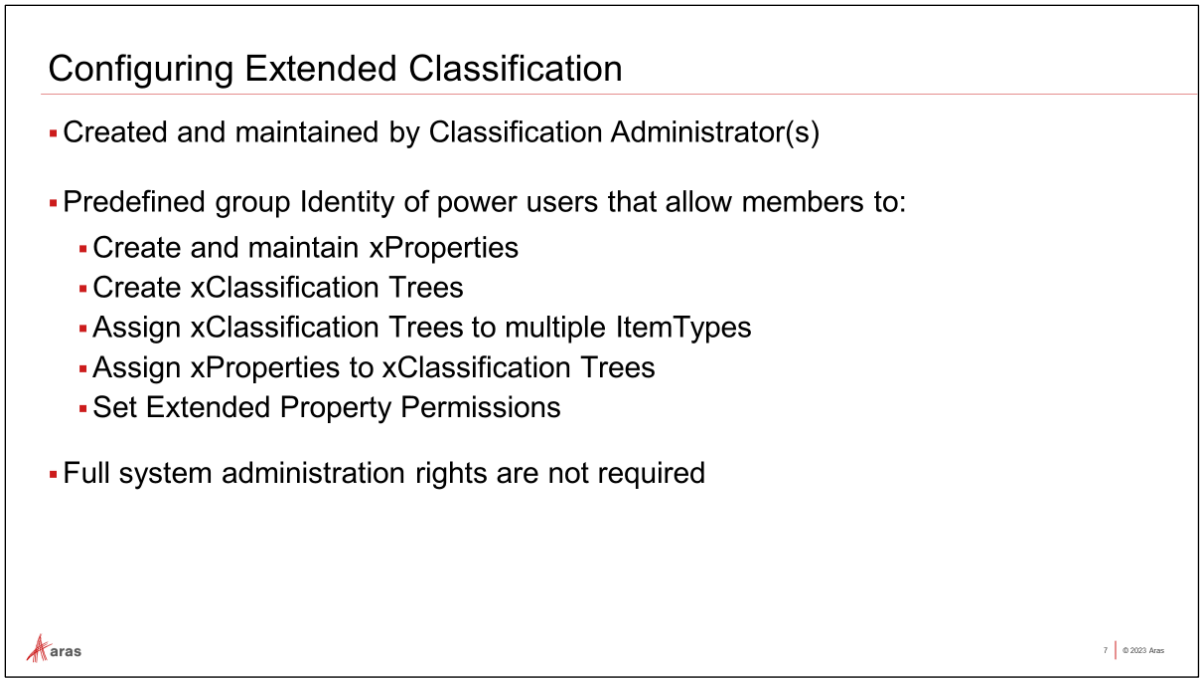
A classification of an item that references associated xProperties for that class.

xClassification Tree

A collection of xClasses arranged in a hierarchical tree (root, branch, leaf). xClassification trees are then assigned to one or more ItemTypes.

xClassPath

Location of an xProperty in an xClassification tree.



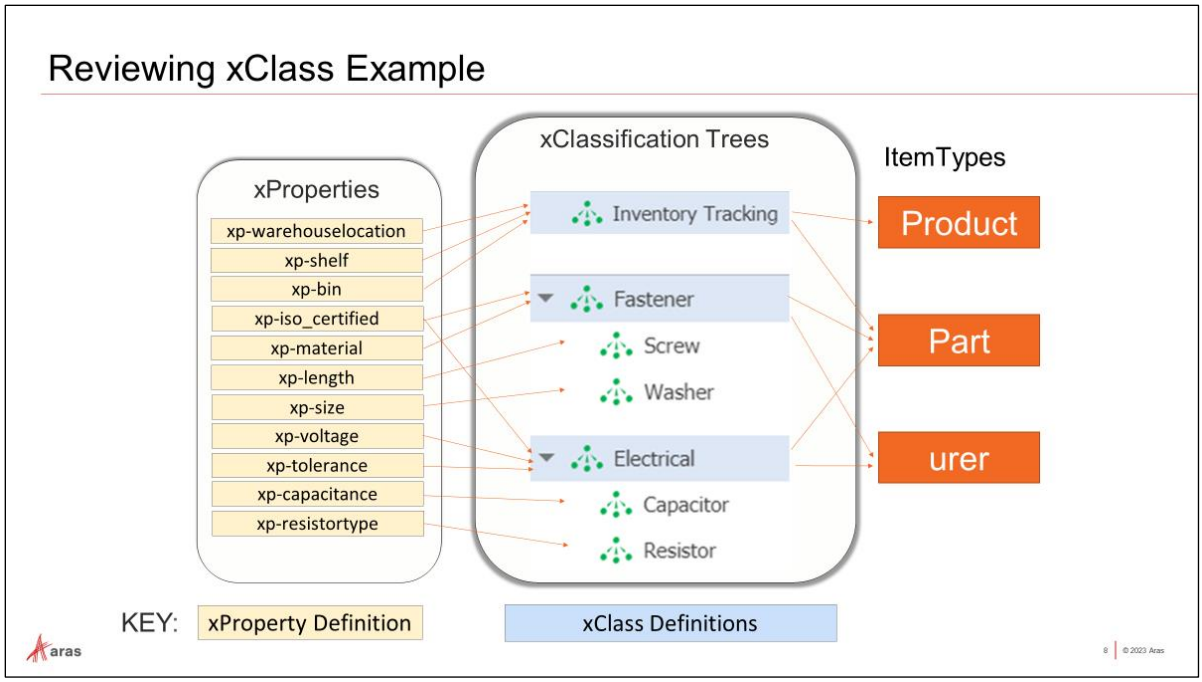
Configuring Extended Classification

Creating and administering xProperties and xClasses requires the user to be a member of the Classification Administrators Identity. Members of this group are not required to have full system administration privileges. Classification Administrators are typically considered power users who have specialization in their industry or field of engineering and an awareness of the attributes needed to define and qualify parts.

Members of this group can specify unique permissions for each xProperty and can also specify who is able to classify or unclassify items.

Try it … Grant Classification Administrator Privileges

1. Login as an Administrator and navigate to Administration > Identities.
2. Edit the Identity named **Classification Administrators,** add *Mike Miller* as Member, and click Done.



Reviewing xClass Example

In this example, you want the ability to classify Parts and Manufacturer Parts using the same xClassification Trees. You will create xProperties and build xClassification Trees, then assign them to the ItemTypes. The **Inventory Tracking** xClassification Tree and xProperties have already been created in the training database.

xClassification Trees

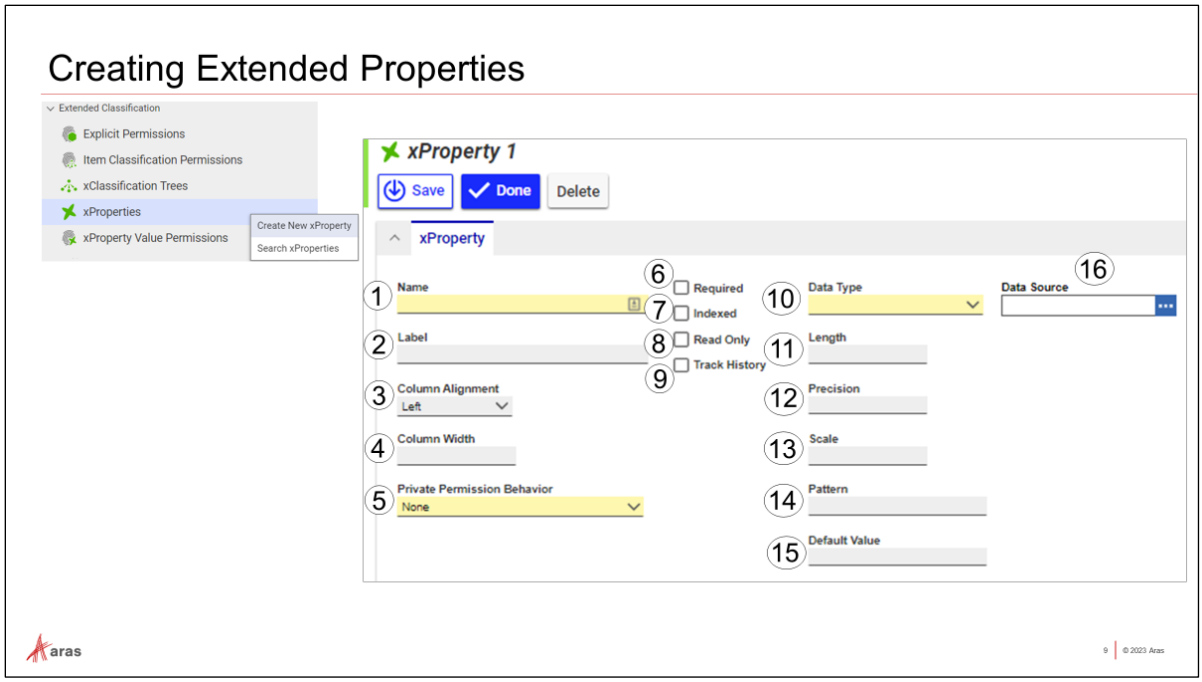
xClassification Trees consist of one or more xClasses that are arranged in a hierarchy. In this example, we will work with three xClassification Trees – **Inventory Tracking**, **Fastener**, and **Electrical**. The Fastener and Electrical xClassification Trees each contain two additional leaf classes, which further categorize these types.

xProperties

xProperties are then assigned to each of the xClasses as shown by the arrows. Note that an xProperty can be assigned to the root level or any leaf xClass level. In this example, xp-voltage is assigned to the root level xClass of the (Electrical) tree, so Capacitor and Resistor will also have access to that property. Note that xProperties can also be shared among different xClasses (e.g., xp-isocertified).

ItemTypes

Finally, xClassification Trees are assigned to one or more ItemTypes to allow end users to classify Items using the provided classifications.



Creating Extended Properties

A Classification Administrator creates Extended Properties (xProperties). Much like standard Properties, xProperties may also be configured with the following attributes:

①	**Name**	The name always begins with the characters xp- to distinguish it from standard Properties.
②	**Label**	The Label text displayed for this xProperty on a Form and in grids.
③	**Column Alignment**	Text alignment of the xProperty value.
④	**Column Width**	Display length of the xProperty value.
⑤	**Private Permission Behavior**	Additional security at the property level.
⑥	**Required**	When set as Required, the value of the xProperty cannot be a Null value in the database.
⑦	**Indexed**	xProperty values are stored in a database table named xp.xPROPERTYVALUES and the database column can be part of a database index to improve query performance.
⑧	**Read Only**	When set as Read Only, the value displays greyed out on the item form and cannot be changed by an end user.
⑨	**Track History**	Property value changes are included in the Item History (audit trail).
⑩	**DataType**	Type of xProperty data (String, Integer, List, Item, etc.).

⑪ **Length** Length of String data.

⑫ **Precision** Decimal – total number of digits.

⑬ **Scale** Decimal – number of places after decimal point.

⑭ **Pattern** Date formats and input formatting.

⑮ **Default Value** xProperty value on new item.

⑯ **Data Source** List/Item datatype source.

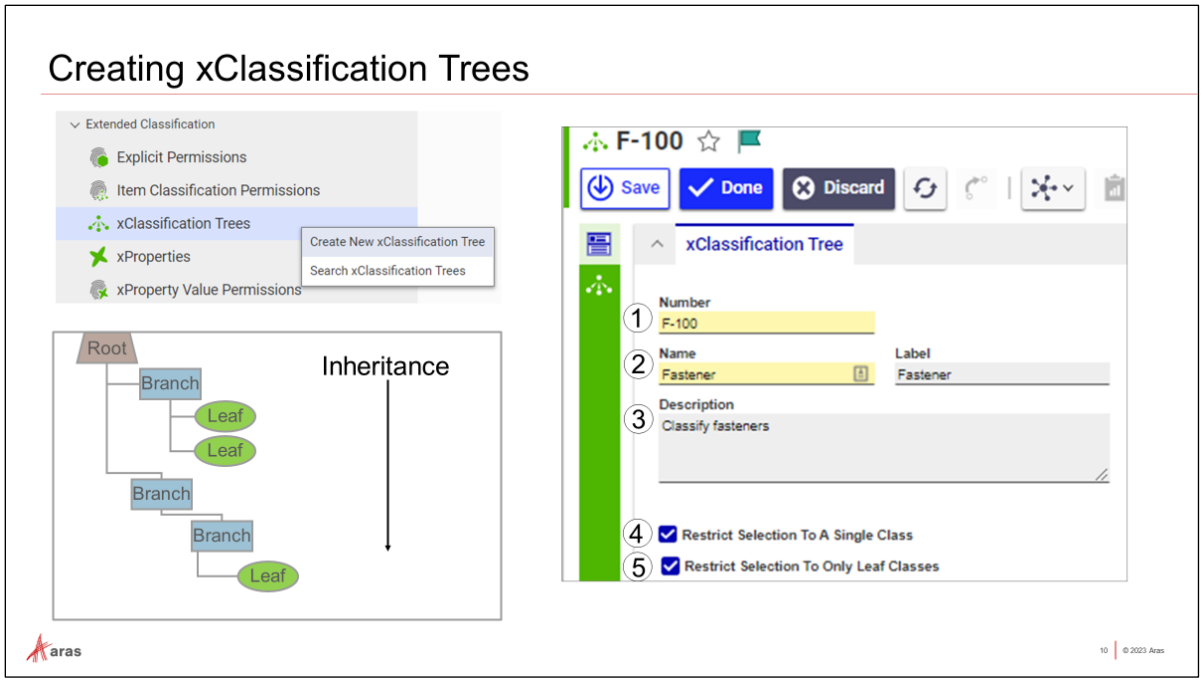
Try it … Create Extended Properties

Please note that the x-property **xp-iso_certified** created here is also required for the lab.

1. Navigate to Extended Classification > xProperties.
2. Create these new Extended Properties providing the following settings:

xProperty Name	Label	Data Type
xp-iso_certified	ISO Certified	Boolean
xp-size	Size cm	Integer

3. Make sure you click Done for each xProperty.



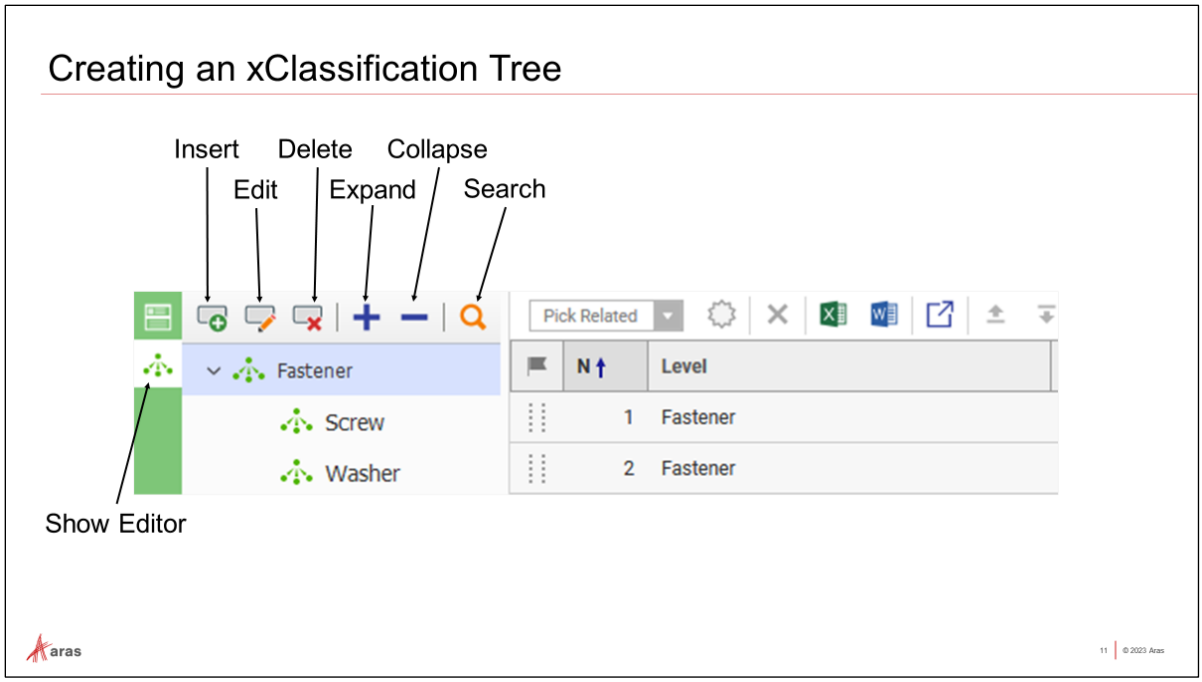
Creating xClassification Trees

An xClassification Tree defines one or more xClasses, which are arranged in a multi-level hierarchy to allow for top-down xProperty inheritance. An xClassification Tree must define one root xClass (top level) and can have multiple leaf xClasses.

A new tree must be saved to the database to display the Editor button in the sidebar.

The xClassification Tree is configured with the following attributes:

①	**Number**	Identifying number for the classification.
②	**Name/Label**	Name and Label text displayed in the UI for this classification.
③	**Description**	Descriptive comments.
④	**Restrict Selection to a Single Class**	When selected, end users can only choose one xClass from this tree for each Item. Disabling this flag allows a user to assign multiple xClasses from this tree to a single Item so that properties from all the selected xClasses are available on the Item.
⑤	**Restrict Selection to Only Leaf Classes**	When selected, end users can only classify an item using a leaf subclass in the xClassification Tree.

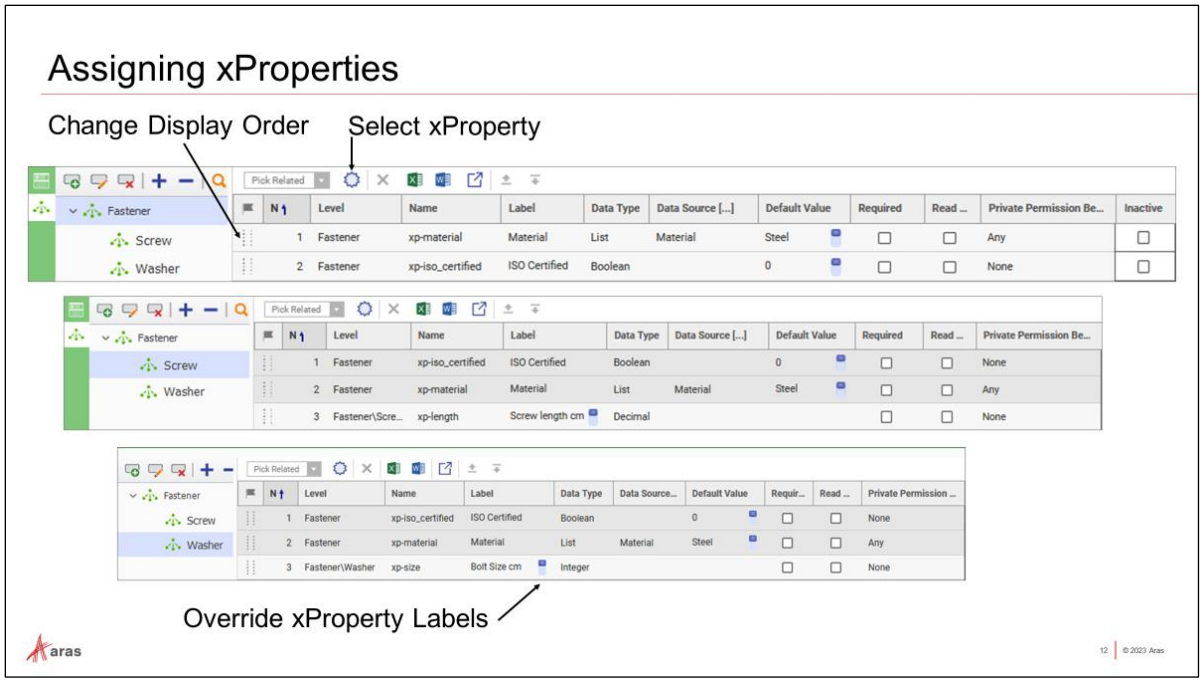


Creating an xClassification Tree

The xClass Tree is created using the xClassification Tree editor, which is visible after clicking Save on the Form. Every tree has a root level xClass with the same name as the xClassification Tree name. From the root level, you can create and manage additional leaf (subclasses) in the tree using the editor.

Try it … Edit an xClassification Tree

1. Navigate to Extended Classification > xClassification Trees and open for editing the **T002** xClassification Tree.
2. On the xClassification Tree definition Form, click the Editor button in the sidebar.
3. Click the Insert toolbar button (or right-click and select Insert Sub-class).
4. Name and label the new sub-class as: *Washer*.
5. If you make a mistake, use the Delete or Edit toolbar buttons to make corrections.
6. Click Save.



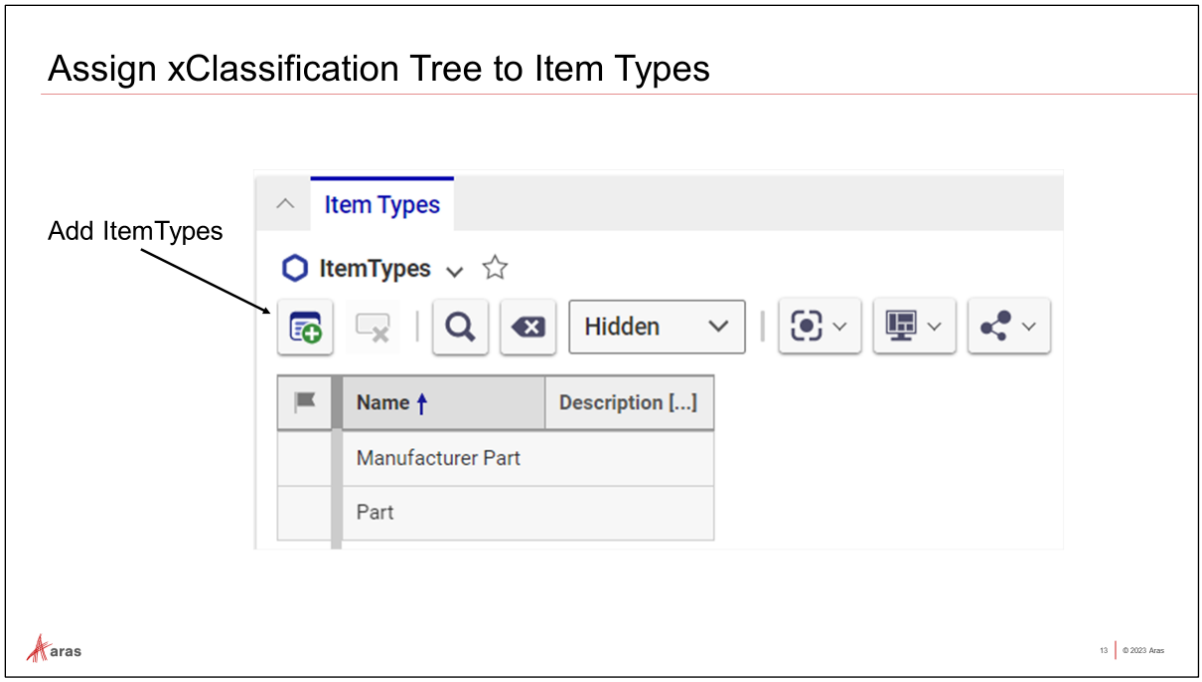
Assigning xProperties

Once the necessary root and leaf xClasses are established, you can assign xProperties to each xClass. Also, you can override certain xProperty values when they are assigned to an xClass. End users see the override xProperty values when they classify an item.

xProperties display in the order they are added to the xClass. To change the display order, click in the Reorder column and drag the row up or down.

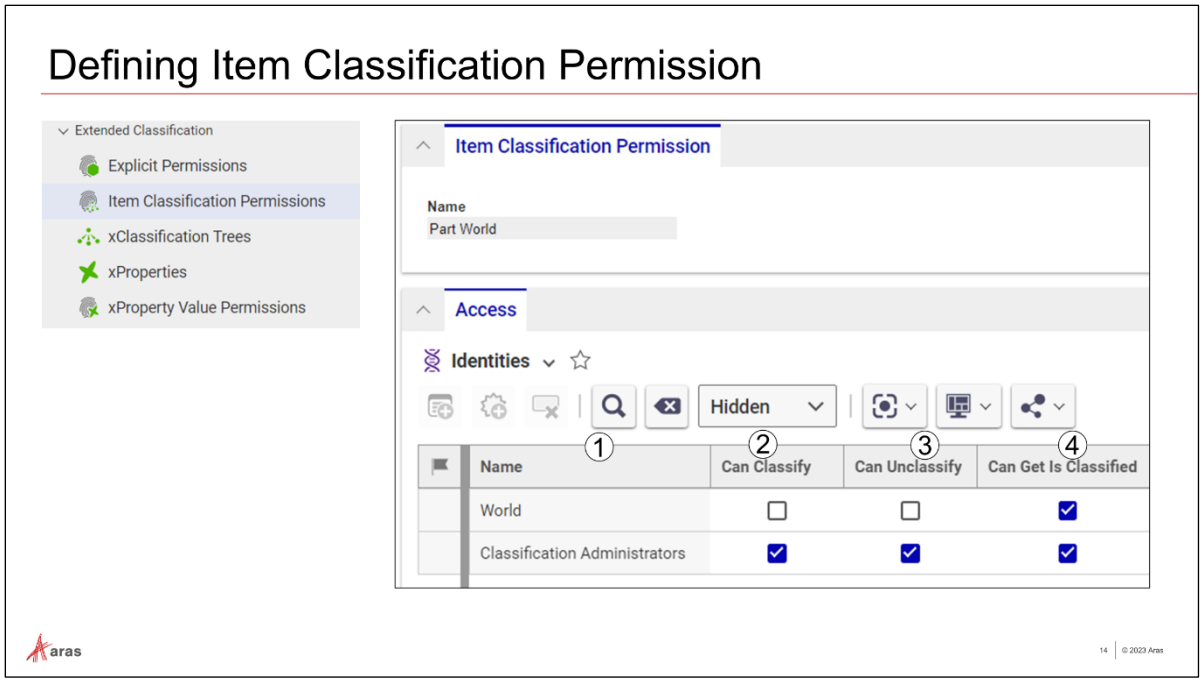
Try it … Assign xProperties

1. Select the **Mechanical** root level xClass, click the New Relationship button , and assign the **xp-iso_certified** xProperty.
2. In the Default Value column, click the Override Switch, and set the default value to 0 (zero).
3. Click Save.
4. Click the **Washer** xClass and notice that the inherited root level xProperties display with a gray background. Click the New relationship button to assign the ***xp-size*** xProperty.
5. Click the Override Switch in the Label column and change the Label of Size cm to *Washer size cm*.
6. Save the x-Classification Tree.



Assign xClassification Tree to ItemTypes

You assign xClassification Trees to one or more ItemTypes. Once assigned, all the properties of the xClassification Tree can be set by the end user creating or editing items of these ItemTypes.



Defining Item Classification Permission

The ability to classify, unclassify and determine if an item is classified is defined by an Item Classification Permission. This permission is assigned to the xClassification Tree for each ItemType associated with the xClassification Tree.

The Access tab of the Permission item contains the following columns:

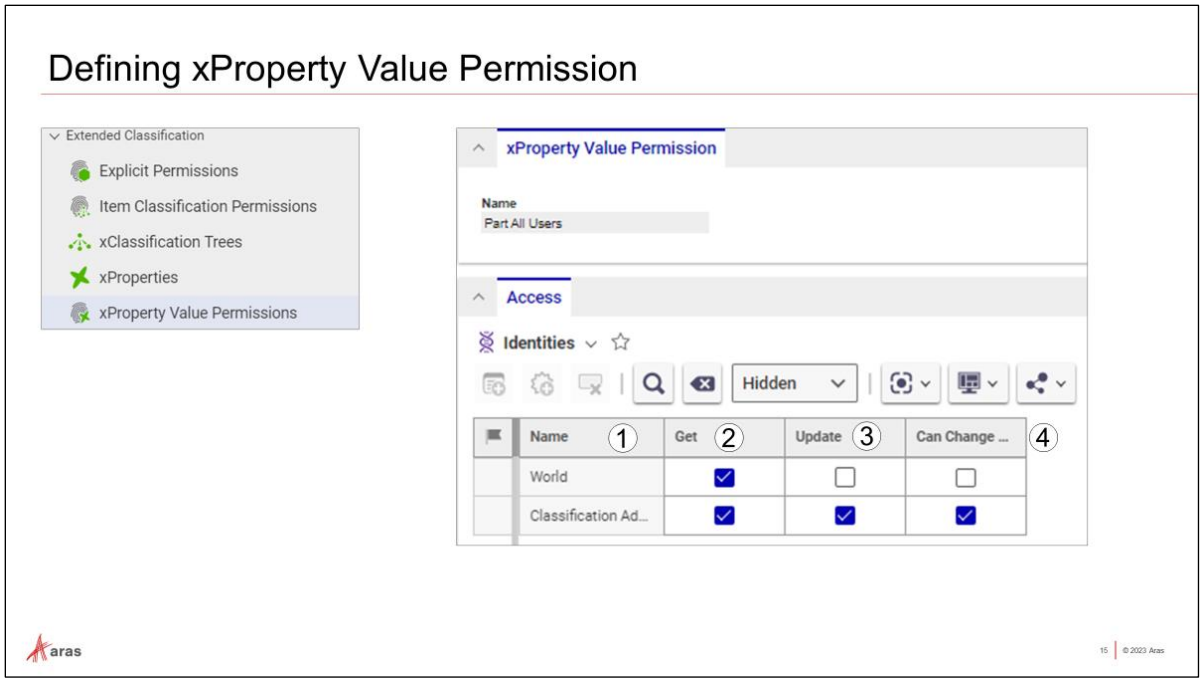
①	**Name**	Identity to assign permission rights.
②	**Can Classify**	Members of the assigned Identity can assign an xClass to an item.
③	**Can Unclassify**	Members of the assigned Identity can unassign an xClass on a previously classified item.
④	**Can Get Is Classified**	Members of the assigned Identity can query if an item is classified with an xClass.

Try it … Change Access for Item Classification Permissions

1. Navigate to Extended Classification > Item Classification Permissions.
2. Create a new permission named **Part World**.
3. In the Access tab, add the following privileges:

Name	Can Classify	Can Unclassify	Can Get Is Classified
World	No	No	Yes
Classification Administrators	Yes	Yes	Yes

4. Click Done.



Defining xProperty Value Permission

Access to xProperty values is controlled by an xProperty Value Permission item. This permission is assigned to the xClassification Tree for each ItemType associated with the xClassification Tree.

The Access tab of the Permission item contains the following columns:

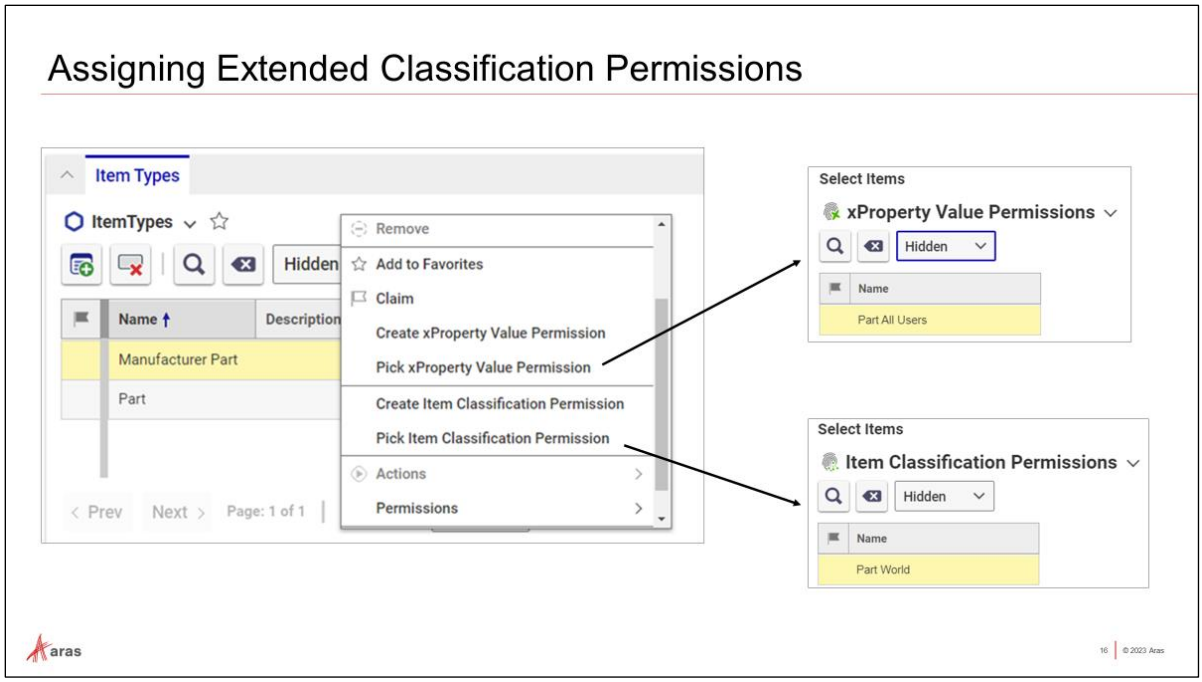
①	**Name**	The Identity to assign permission rights.
②	**Get**	Members of the assigned Identity can see the value of an xProperty. The word Restricted appears in the place of the value if a user does not have Get access.
③	**Update**	Members of the assigned Identity can assign or change the value of an xProperty.
④	**Can Change Access**	Members of the assigned Identity can set a private permission on an xProperty.

Try it … Change Access for an xProperty Value Permission

1. Navigate to Extended Classification > xProperty Value Permissions.
2. Create a new permission named **Part All Users** for editing.
3. In the Access tab, add the following privileges:

Name	Get	Update	Can Change Access
World	Yes	No	No
Classification Administrators	Yes	Yes	Yes

4. Click Done.



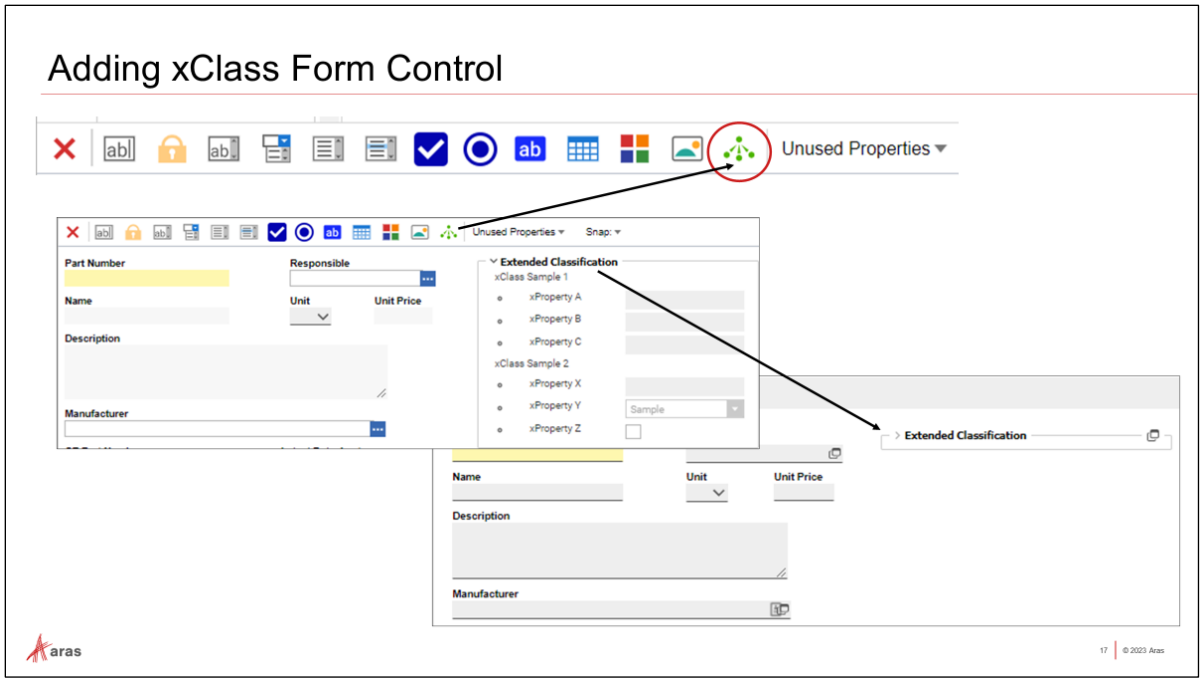
Assigning Extended Classification Permissions

xProperty Value Permissions and xClassification Permissions are assigned using the context menu on the Item Types tab of an xClassification Tree. Each ItemType may have a different set of Permissions. The context menu can be used to select an existing Permission or create a new one.

Try it … Assign Extended Classification Permissions

In this exercise, you select existing permission sets.

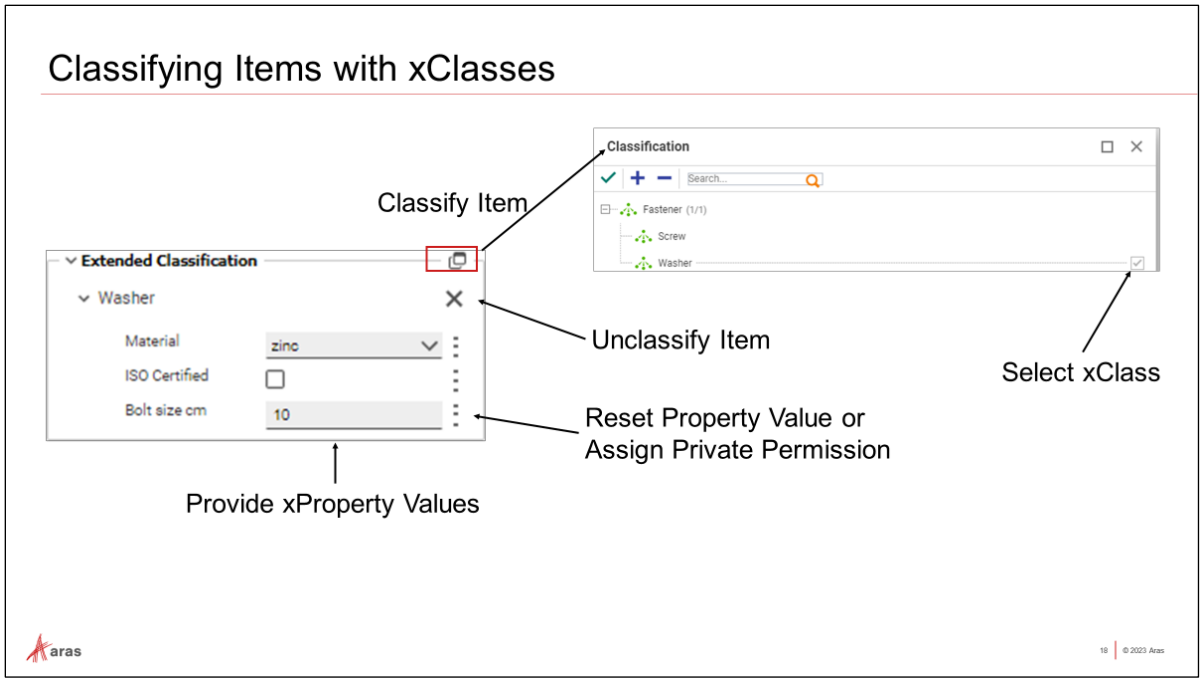
1. Open the **T002** *Mechanical* xClassification Tree for editing.
2. Click on the Form sidebar button.
3. In the Item Types relationship tab, right-click on the **Manufacturer Part**, select *Pick xProperty Value Permission* from the context menu, and choose the **Part All Users** permission.
4. Again, right click on **Manufacturer Part**, select *Pick Item Classification Permission* from the context menu, and choose **Part World**.
5. Click Done.



Adding xClass Form Control

A reserved xClass control is available in the Form editor toolbar to allow users to classify, unclassify, and assign property values. Editing the Form requires full admin rights and is not part of the scope for a Classification Administrator.

The Form editor shows the xClass control as a representation of what the control will look like to an end user to help you with placement. The actual size of the control differs depending on the number of configured xClasses for the ItemType.



Classifying Items with xClasses

Once an ItemType has been configured to support xClasses, an end user can classify an item based on their permissions. If a Classification Administrator modifies the xProperty definitions and/or xClassification Trees after they have been assigned to Items, those changes are propagated to the existing items.

The available xProperties display in the xClass control on the Form to allow data entry of values.

To Unclassify an Item

To unclassify an Item, lock an item for editing and click the Unclassify Item icon (X) to remove the classification.

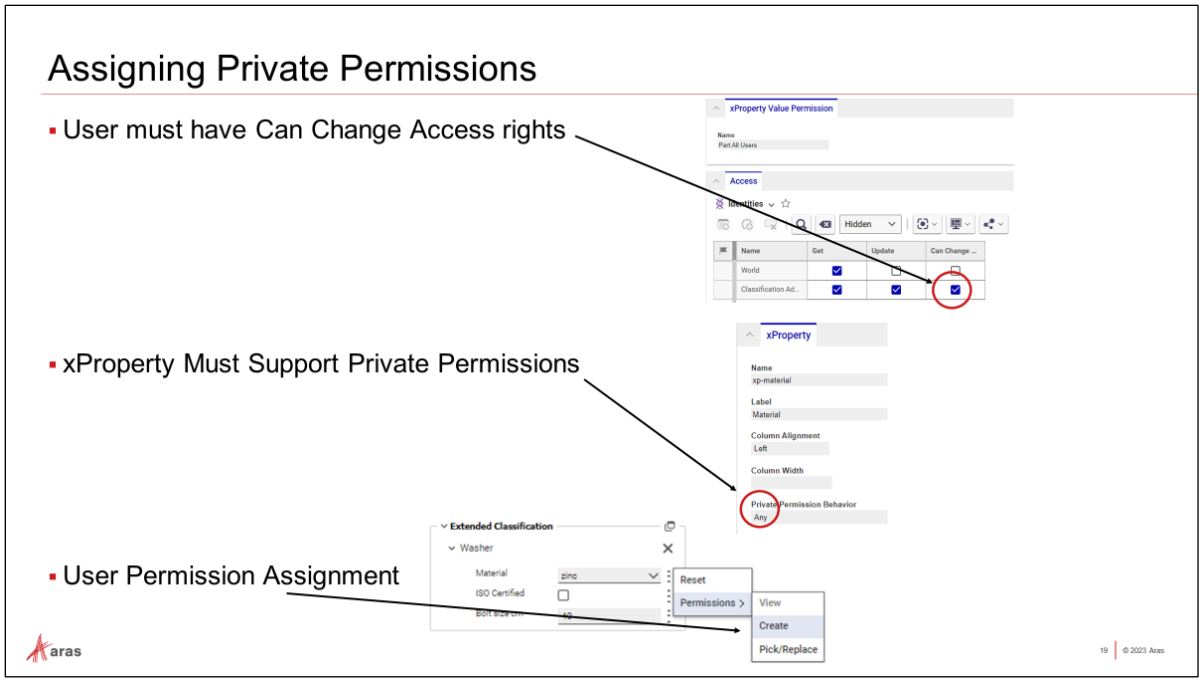
To Reset Values or Assign a Private Permission

To Reset xProperty values or Assign a Private Permission, click the icon (three vertical dots) to the right of an xProperty to reset the value back to its default value. (The default value could be empty.)

Once reset, you can also assign a Private xProperty Value permission.

Try it … Classify an Item

1. Navigate to Sourcing > Manufacturer Parts and create a new Manufacturer Part.
2. Enter *FPW-10* as the Part Number and *Flat Washer* as the Name.
3. In the xClass control, click the Classify Item icon to display the available xClasses.
4. Click the Expand All button to display the entire xClassification Tree.
5. Select **Washer** and click the green checkmark.
6. Expand the Washer xClass, and enter *10* for the Washer size cm.
7. Click Done.



Assigning Private Permissions

Access to xProperty values is normally configured using xProperty Value permissions that are assigned to an xClassification Tree. An xProperty inherits the Permissions from the xClass to which it is attached unless a different xProperty Permission is set.

To allow an xProperty to have a Private Permission, both of the following configurations must be set.

- The end user must be granted the Can Change Access right on an xProperty Value Permission item assigned to the ItemType in the xClassification Tree.
- The xProperty must also be configured with the Private Permission Behavior of **Any**.

Try it … Allow Private xProperty Permissions

1. Navigate to Extended Classification > xProperties and open the **xp-qa_approval_authority** xProperty for editing.
2. In the Private Permission Behavior field, choose **Any**.
3. Click Done.

Try it … Assign a Private xProperty Permission

Add a private permission for the Material xProperty on the Manufacturer Part Number FPW-10.

1. If the **FPW-10** Manufacturer Part is opened in a tab, close this tab. This will allow the previous configuration to take effect on the item.
2. Navigate to Sourcing > Manufacturer Parts and open for editing the **FPW-10** Flat Washer.
3. Expand the Washer xClass, click the selection button to the right of the **Quality Approval Authority** xProperty, and select Permissions > Create from the context menu.
4. Name the xProperty Value Permission: *QAA-PP*.

5. We want Engineering to be able to both Get and Update the *Quality Approval Authority* xProperty value for this Manufacturer Part. Add the Identities, and set the following values for this new xProperty Private Permission:

Name	Get	Update	Can Change Acc...
Classification Administr...	☑	☑	☑
Engineering	☑	☑	☐
World	☑	☐	☐

6. Click Done.
7. Return to the (**FPW-10**) Manufacturer Part and click Done to save the new private permission relationship.

Try it… Test Private xProperty Permissions

Test the private permission by logging in as members of different group identities.

1. Using a different browser, log in as Jane Dow, who is not a member of a group with permission to update the xProperty values.
2. Find and open for editing the Manufacturer Part **FPW-10**, Flat Washer.
3. Expand the Washer xClass, change the Quality Approval Authority value, and click Done.
4. Dismiss the message window, click the Discard button, and log out.
5. Log in as Mike Miller, member of the Engineering group, and open for editing the Manufacturer Part **FPW-10**, Flat Washer.
6. Change the Quality Approval Authority value.
7. Click Done and logout.

Summary

In this unit, you learned how Extended Properties and Extended Classes are used to classify an item.

You should now be able to:

- Define Extended Properties and Classes
- Explain the responsibilities of the Classification Administrator
- Create Extended Properties and Classes
- Create and Manage Extended Classification Trees
- Set an xClass/xProperty private Permission.

Review Questions

1. What are the benefits of using Extended Properties and Classes?

2. How do you configure a user to be a Classification Administrator?

3. What are the responsibilities of the Classification Administrator?

4. What are two permissions that must be configured for each ItemType associated with a Classification Tree?

5. An xProperty name must always begin with what characters?

Lab Exercise

Goal

Be able to create xClasses and xProperties to support Extended classification.

Scenario

In this exercise, you create several new xProperties and assign them to a new xClassification Tree. You then assign the xClassification Tree to the Part and the Manufacturer Part ItemTypes.

Steps

1. As an Administrator, create a new List item named *Resistor Types*, provide the following entries, and click Done.

Label	Value
Fixed	Fixed
Variable	Variable
Thin Film	Thin Film

2. Navigate to Extended Classification > xProperties and create these new Extended Properties:

Property Name	Label	Data Type	Data Source	Precision	Scale
xp-voltage	Voltage	Decimal		6	2
xp-tolerance	Tolerance	Decimal		6	3
xp-resistor_type	Resistor Type	List	Resistor Types		

Note: Please create the x-property named xp-iso_certified if you did not build it with the instructor earlier.

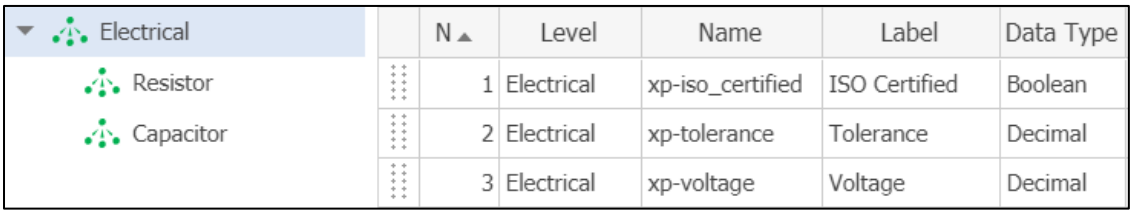
Property Name	Label	Data Type	Data Source
xp-iso_certified	ISO Certified	Boolean	

3. Create a new xClassification Tree using the following settings:

Number	E-100
Name	Electrical
Label	Electrical
Description	Classification Tree used for Parts and Manufacturer Parts
Restrict Selection to Only Leaf Classes	Checked
Restrict Selection to a Single Class	Checked

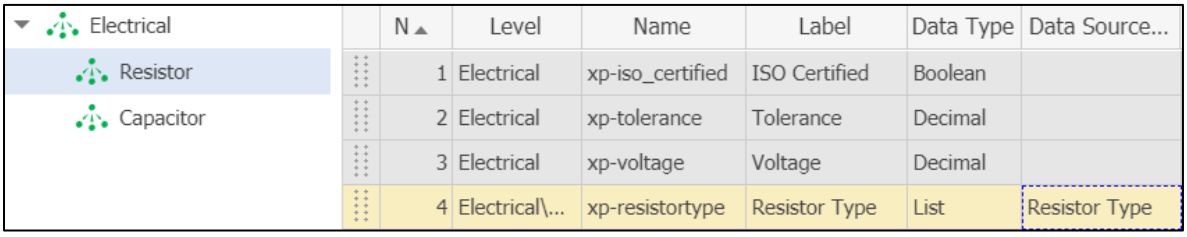
4. Save the configuration.
5. Click the Editor sidebar button and create two new Sub-classes in the Electrical xClassification Tree named *Resistor* and *Capacitor*.

6. Using the xClassification Tree Editor, assign the **xp-voltage, xp-tolerance,** and **xp-iso_certified** xProperties to the root **Electrical** xClass.



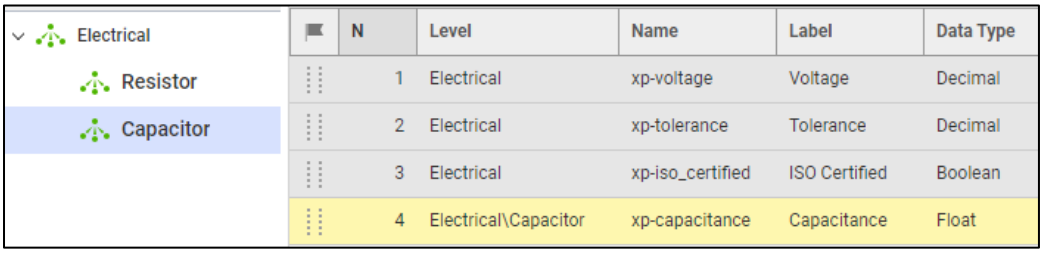
Electrical	N	Level	Name	Label	Data Type
Resistor	1	Electrical	xp-iso_certified	ISO Certified	Boolean
Capacitor	2	Electrical	xp-tolerance	Tolerance	Decimal
	3	Electrical	xp-voltage	Voltage	Decimal

7. Assign the **xp-resistor_type** xProperty to the **Resistor** sub-class.



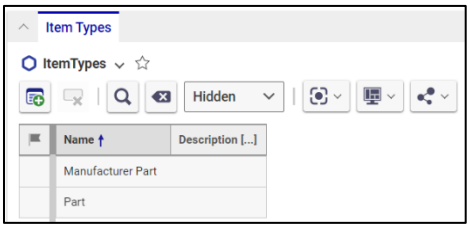
Electrical	N	Level	Name	Label	Data Type	Data Source...
Resistor	1	Electrical	xp-iso_certified	ISO Certified	Boolean	
Capacitor	2	Electrical	xp-tolerance	Tolerance	Decimal	
	3	Electrical	xp-voltage	Voltage	Decimal	
	4	Electrical\...	xp-resistortype	Resistor Type	List	Resistor Type

8. Assign the **xp-capacitance** xProperty to the **Capacitor** sub-class.



Electrical	N	Level	Name	Label	Data Type
Resistor	1	Electrical	xp-voltage	Voltage	Decimal
Capacitor	2	Electrical	xp-tolerance	Tolerance	Decimal
	3	Electrical	xp-iso_certified	ISO Certified	Boolean
	4	Electrical\Capacitor	xp-capacitance	Capacitance	Float

9. Click the Form sidebar button to display the xClassification Form and add the **Part** and **Manufacturer Part** ItemTypes to the ItemTypes relationship grid.



10. Right click on each ItemType to pick both the Item Classification Permission and xProperty Value Permission. Select the existing **Part All Users** xProperty Value permission and **Part World** Item Classification permission and apply to each ItemType.
11. Click Done.
12. Create new Parts/Manufacturer Parts and classify them with the new xClass /Sub-class.

This page intentionally left blank.

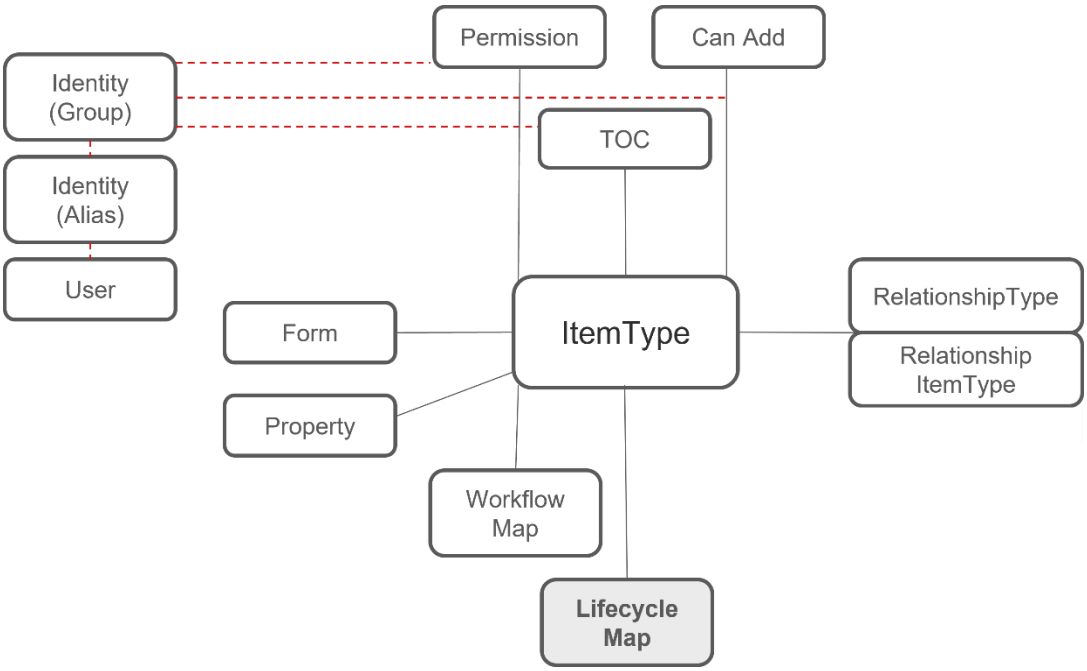
Unit 13 Defining Lifecycle Maps and Versioning

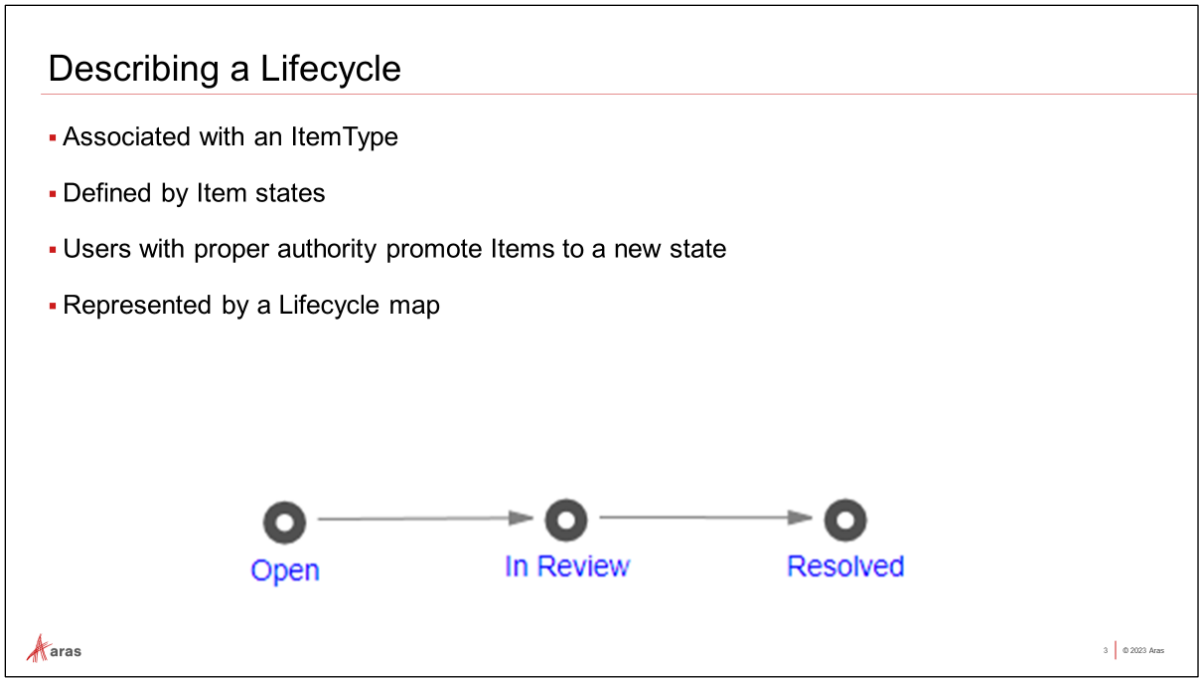
Overview

In this unit, you learn how to define Lifecycle maps for an ItemType and how to control how a user promotes an Item through a Lifecycle. You also learn how to control Item versions and how to configure different behaviors when related Items are versioned.

Objectives

- Identify the Lifecycle States
- Define Lifecycle Transitions
- Create a Lifecycle Map for an ItemType
- Define Lifecycle State Permissions
- Promote an Item through Lifecycle States
- Configure and View Item Versions
- Define Versioning Behavior





Describing a Lifecycle Map

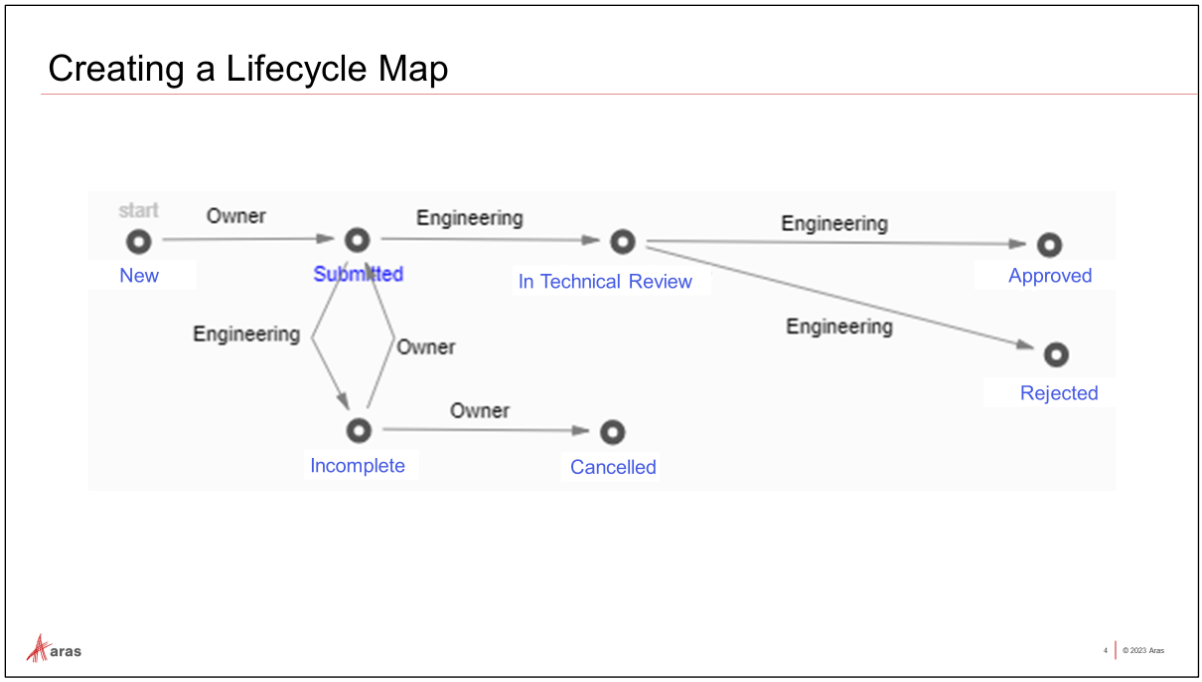
A Lifecycle map is a series of states (i.e., stages, gates, or milestones) that an Item traverses during its existence. Most business processes define high-level stages to track the progress of an object in a Lifecycle map.

An item can only have a single state at any given time. The example above uses three Lifecycle States.

Please note that it is important to control permissions access for each Lifecycle State.

Every Lifecycle map has a Starting state and Transitions that define the connection between two states, the promotion path.

Each Lifecycle Transition includes a role: The Identity that can promote to the next Lifecycle State.



Creating a Lifecycle Map

There are two steps to configure a Lifecycle map for an ItemType:

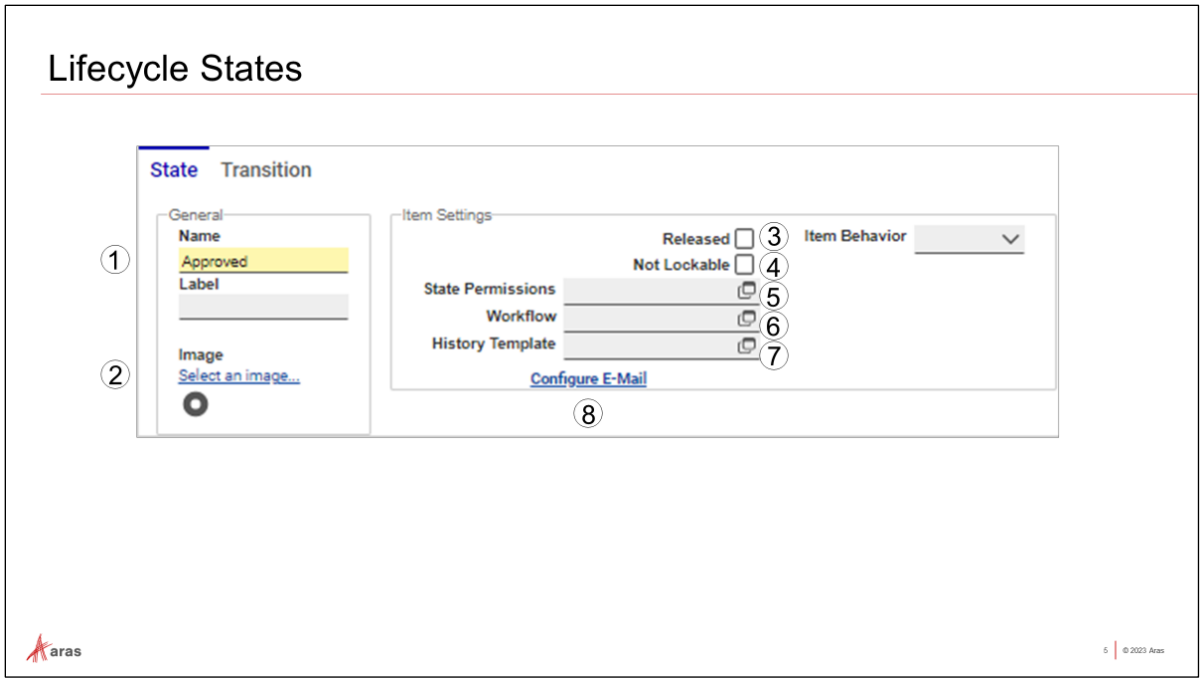
- Define the Lifecycle Map
- Assign the Lifecycle Map to the ItemType

Try it … Create and Configure an ItemType Lifecycle Map

1. Login as an Administrator (your student login) and navigate to Administration > Life Cycle Maps.
2. Create a new map and enter *Design Request* in the Name field.
3. In the Map canvas, right-click and add the following States to the Lifecycle map:

New (renamed from Start)
Submitted
In Technical Review
Approved
Rejected
Incomplete
Cancelled

4. Click Save.



Lifecycle States

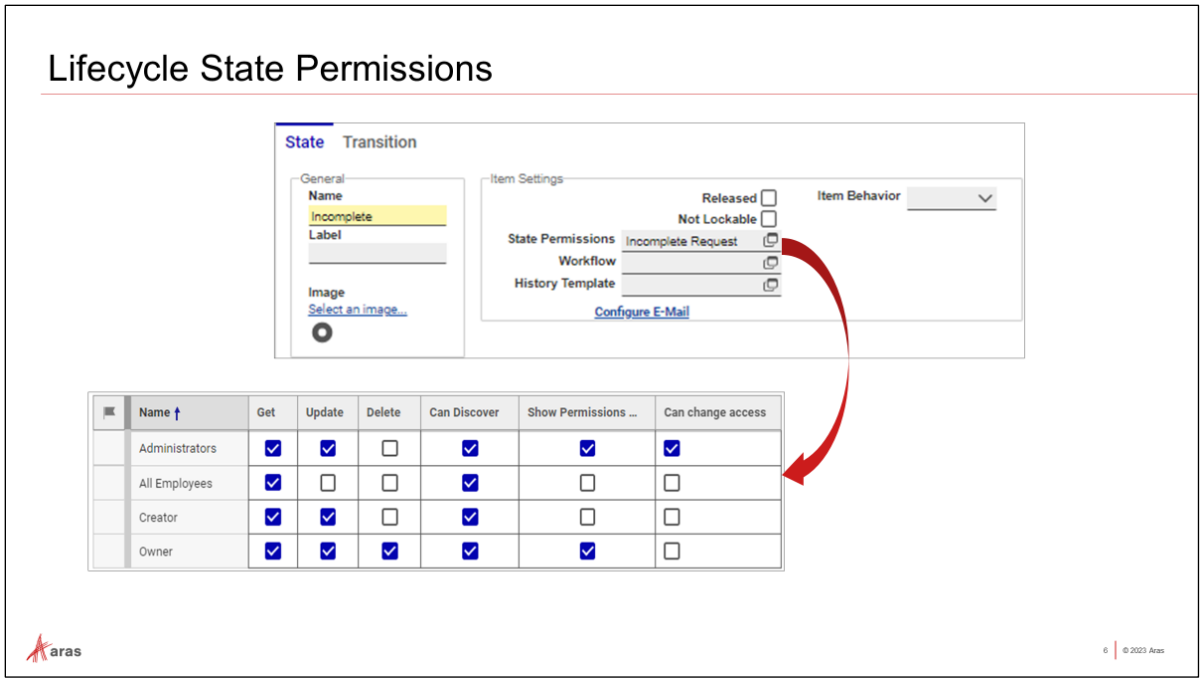
Items must go through all the appropriate Lifecycle states as designed in the Lifecycle map. The following fields define a Lifecycle State:

①	**Name**	The Name of the State in the Lifecycle map. If you supply a label, it displays to the user instead of the name and can be localized for foreign languages.
②	**Image**	A Custom image for the state.
③	**Released**	Signifies the Item is released and should be revised to the next major revision if versioning is supported for the Item. The Lifecycle map is also reset to the first Lifecycle state.
④	**Not Lockable**	The Item is not editable while in this State.
⑤	**State Permissions**	State Permissions overrides any default ItemType Permissions and xClass/xproperties permissions while in this State. The permission persists until another state sets another permission. State permissions must also appear as allowed permissions on the ItemType definition to work correctly.
⑥	**Workflow**	The State starts a Workflow.
⑦	**History Template**	Allows the History Template defined on the ItemType to be overridden by this defined template. For the override to work correctly the original ItemType must have a History Template defined.
⑧	**Configure E-Mail**	Allows email messages to be sent when this state becomes current on an Item.

Try it … Set Specific State Settings

In this exercise, you set the **In Technical Review** state to **Not Lockable** to prevent editing and set the Approved state to **Released**.

1. Select the **In Technical Review** state and in Item Settings, click to select the Not Lockable checkbox.
2. Select the **Approved** state and in Item Settings, click to select the Released checkbox.
3. Click Save.



Lifecycle State Permissions

If desired, each state in a Lifecycle map can have a different permission set. Different permission sets guarantee that appropriate access to all items can be set as items progress through a lifecycle process or are promoted to a state that should have restricted access.

Carefully consider the permissions for each Lifecycle state. If there are no State permissions assigned, the permissions from the previous state continue to be in effect.

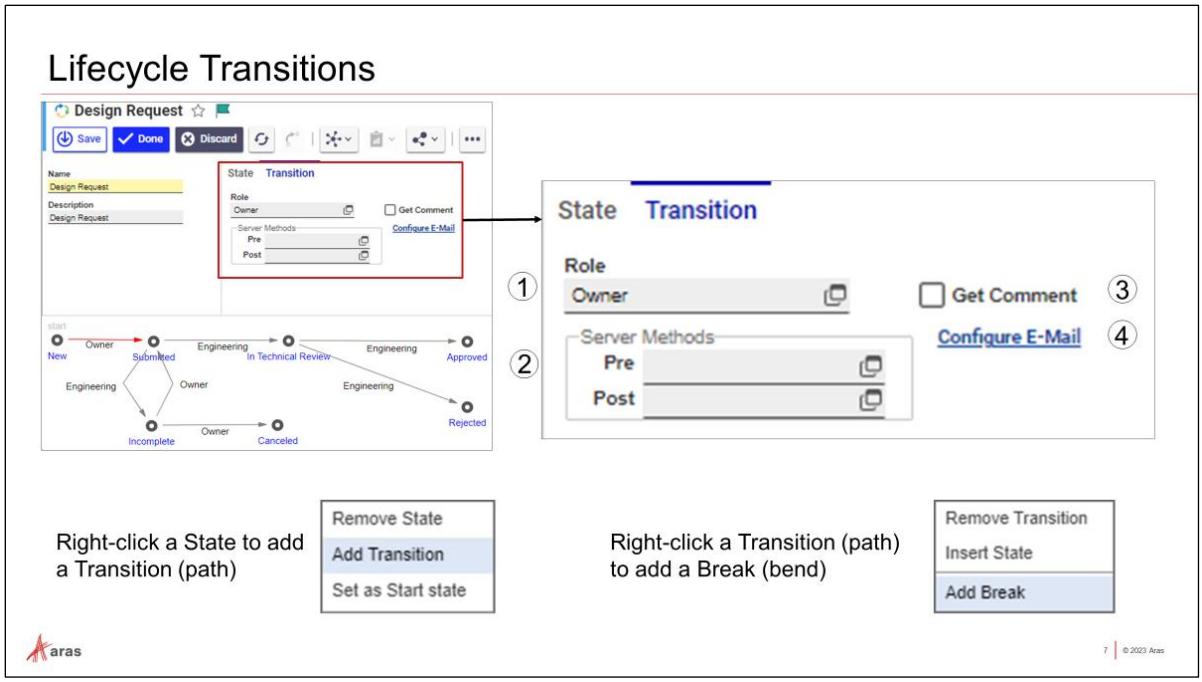
Try it … Configure a Permission on a Lifecycle State

In the Design Request solution, Engineering can promote a request to the Incomplete state when there is missing information on the form. At Incomplete, we want to make sure that the owner can input the missing information, but that All Employees can only view the form. If an issue arose so that the owner was not available, the Administrators group should be able to change permissions for a single request.

1. Navigate to Administration > Permissions and create a new permission named *Incomplete Request*.
2. Set the permissions as shown here and click Done.

Name	Get	Update	Delete	Can Discover	Show Permissions ...	Can change access
Administrators	☑	☑	☐	☑	☑	☑
All Employees	☑	☐	☐	☑	☐	☐
Creator	☑	☑	☐	☑	☐	☐
Owner	☑	☑	☑	☑	☑	☐

3. Return to the **Design Request** Lifecycle map.
4. Select the **Incomplete** state in the map canvas and in the Items Settings area, choose the new **Incomplete Request** permission for the State Permission field.
5. For all other Lifecycle states, set the State Permissions to **Design Request** permission.
6. Click Save.



Lifecycle Transitions

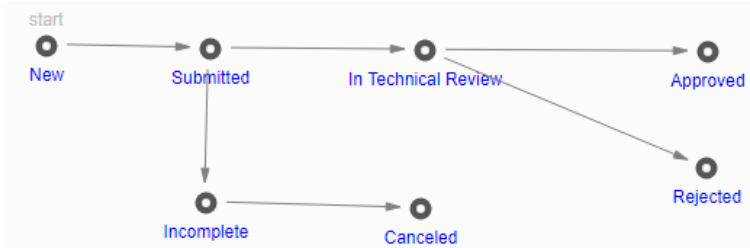
The following fields define a Lifecycle Transition:

①	**Role**	The Identity that can interact with this Transition (Promote).
②	**Server Methods**	Custom server Methods you create to provide custom behavior before and after this transition.
③	**Get Comment**	Presents a dialog box to the end-user when a promotion occurs to capture user remarks.
④	**Configure E-Mail**	Configures message(s) to send when this transition occurs.

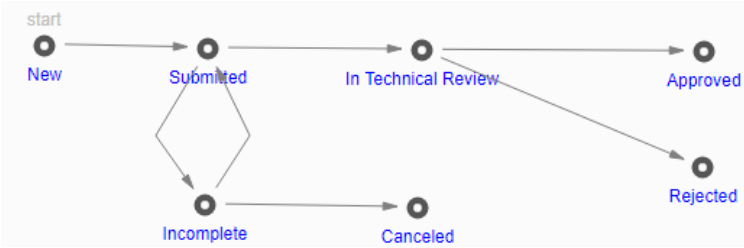
Try it … Create Transitions and Provide Roles

1. Connect each State with a Transition, as follows:
   a. Right-click on a state (the from state) and select **Add Transition**.
   b. Move the mouse pointer on the next state (the To state) and click.

   This action adds a transition from the From state to the selected To state.



2. Add a breakpoint in the Transitions between the **Submitted** and the **Incomplete** States to make the diagram mode readable.
   a. To add a break, right-click on a transition arrow and select **Add Break**.
   b. Drag the transition line as required.



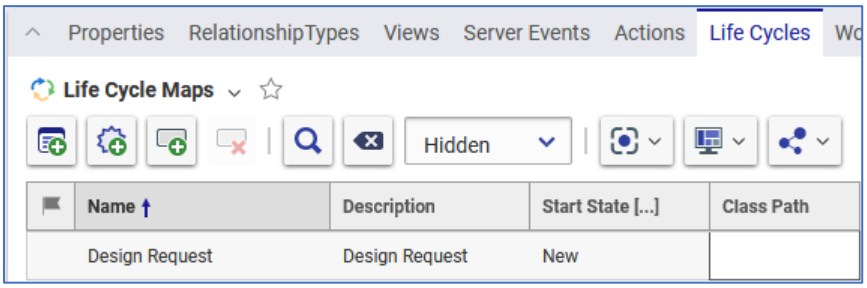
3. Add another Transition from Incomplete to Submitted.
4. Provide the Role identity for each Transition, as follows.

From State	To State	Role
New	Submitted	Owner
Submitted	Incomplete	Engineering
Incomplete	Submitted	Owner
Incomplete	Cancelled	Owner
Submitted	In Technical Review	Engineering
In Technical Review	Approved	Engineering
In Technical Review	Rejected	Engineering

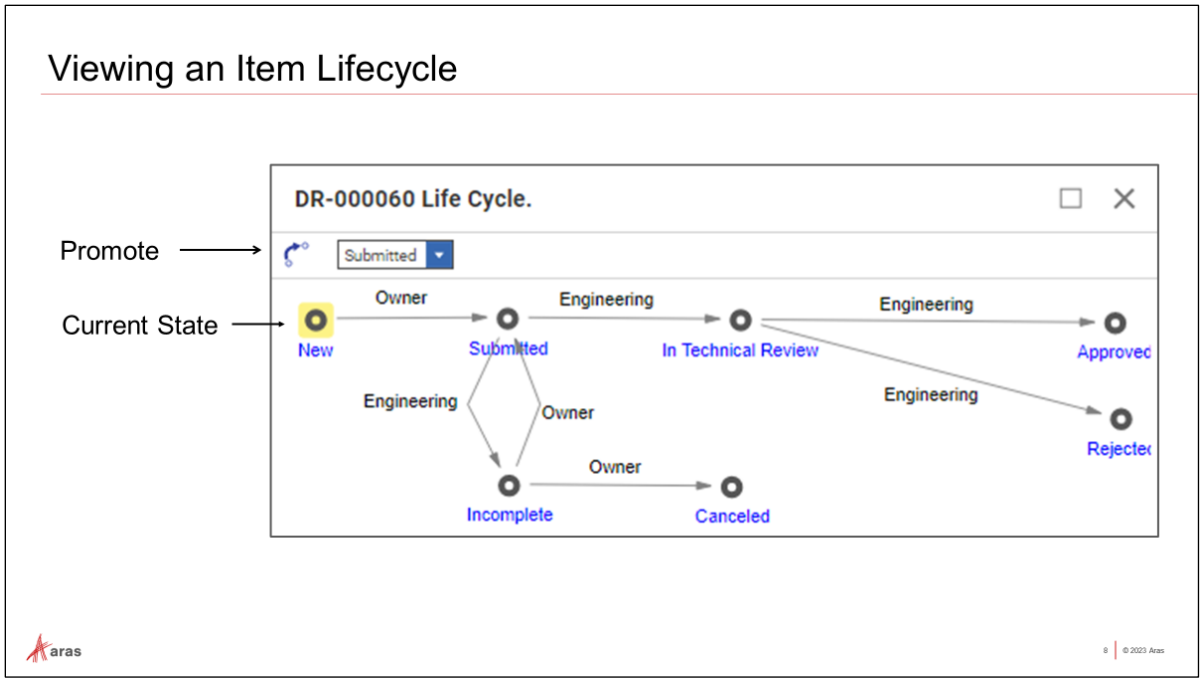
5. Click Done.

Try it … Associate an ItemType with a Lifecycle Map

1. Open the **Design Request** ItemType for editing.
2. Click the Lifecycles relationship tab and add the **Design Request** Lifecycle map.
3. Click Done.



A Lifecycle map can use the classification subclasses if the ItemType is configured with a Class Structure. The Class Path value represents the classification subclass.



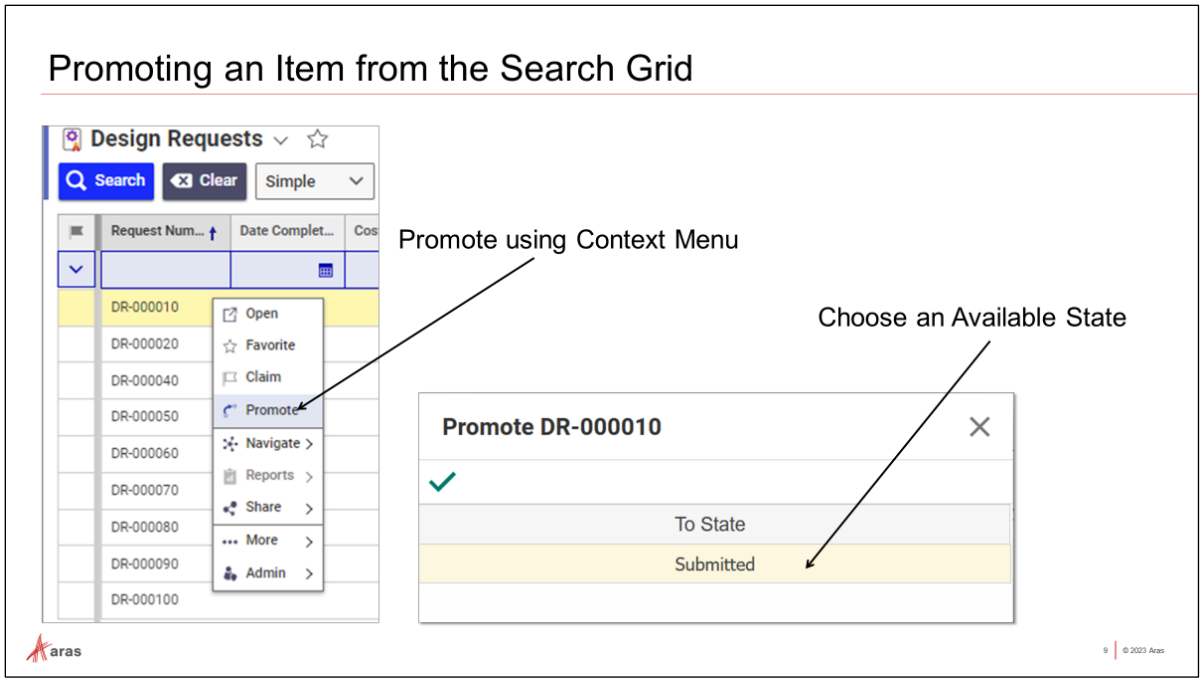
Viewing an Item Lifecycle Map

To view an Item Lifecycle map, open an Item, and select Life Cycle from the Navigate menu.

Try it … Create a New Item and View the Lifecycle Map

1. Logged in as an Administrator (your student login), navigate to Product Management > Development > Design Requests, create a new Design Request item, set Administrators as the Owner, click the Done button, and note the Design Request number.
2. Click the Navigate button and select **Life Cycle** to view the Lifecycle map.

   Notice the current state **(New)**.
3. Promote the Item to the next Lifecycle state by choosing the **Submitted** state from the drop-down list (if not already selected) and clicking the Promote button.
4. The State field on the form now displays Submitted.
5. Using a different browser, sign in as an Engineer (e.g., **Diane Prescott**, **Peter Smith, John Young, or Susan Harris**), open the Design Request for viewing, and continue promoting the Item until it reaches its end state **(Approved)**.
6. Sign out.



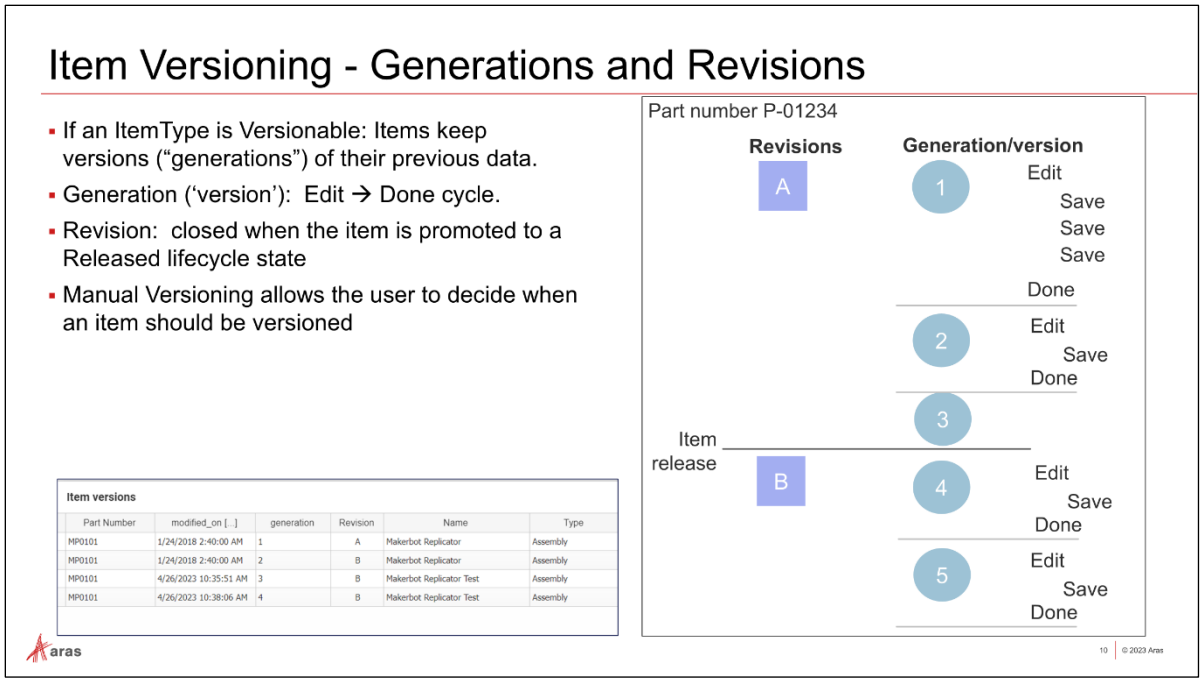
Promoting an Item from the Search Grid

You can use the context menu (right mouse click) to promote an Item from the Search Grid.

Try it … Promote a Design Request to Incomplete

In this exercise, create a Design Request to demonstrate only the Owner can promote it to Submitted and edit the item in the Incomplete state.

1. Sign in as **Jane Dow** (jdow / innovator), who is not a member of Engineering. Create a New Design Request item, set Jane Dow as the Owner, and click Done.
2. Promote the Design Request to **Submitted**, and then attempt to promote it to **Incomplete**.
3. Sign out and sign back in as an Engineer (e.g., Diane Prescott, **Peter Smith, John Young,** or **Susan Harris**) and promote the Design Request item to the **Incomplete** state.
4. Attempt to open the Design Request for editing.
5. Leave the Design Request in the **Incomplete** state, sign out, and close the second browser.



Item Versioning - Generations and Revisions

major_rev is the Version Property used for major changes. It is controlled by business rules that cause the Version to increment when changes are made to an Item in its Released state.

minor_rev is reserved for custom programming.

Generation (version) is an Integer Property that automatically increments from Aras Innovator core processes and rules. A new generation is created whenever an Item Is Edited and Saved. When a user clicks on Done, no additional edits can be made to the current generation.

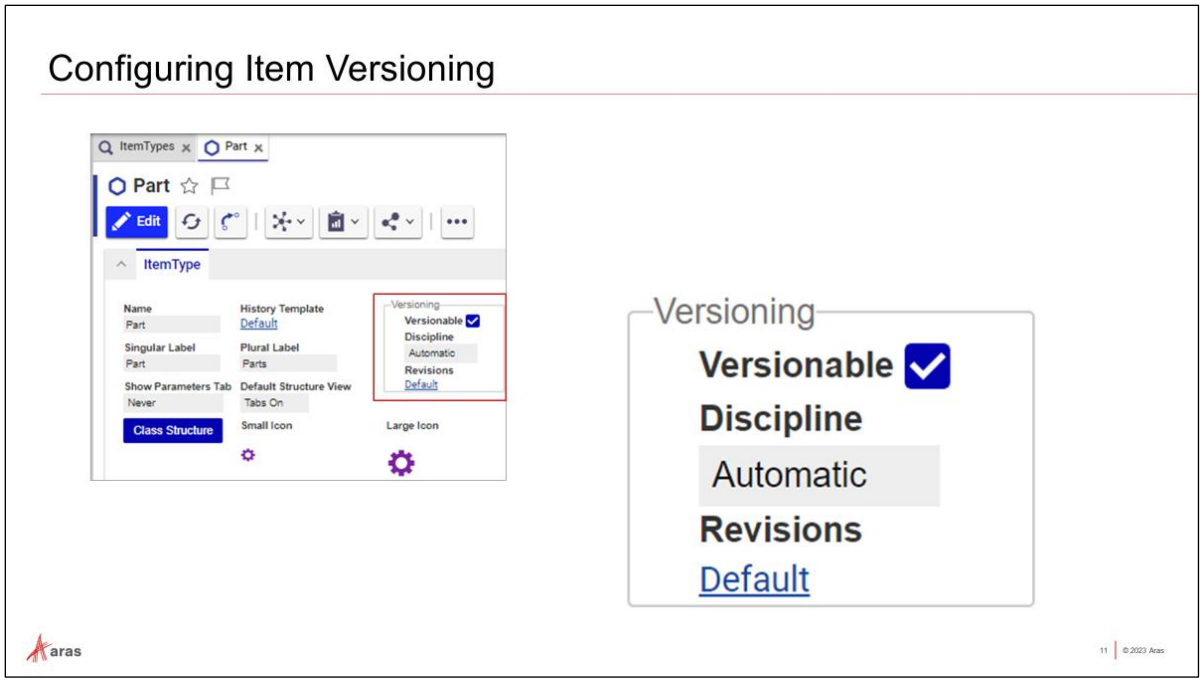
Lifecycle map is a customer-defined sequence of States and Transitions used as a primary driver for business rules. All Items are defined to be in one State within their specified Lifecycle map. Flags defined for each State determine Item behaviors.

Lifecycle Start State is the State within the Lifecycle map that has specifically been designated as the starting point for the Lifecycle map. When a new Item instance is created, it is automatically set to the Start State. When Versioning is enabled, the Item is set back to its Lifecycle Start State when its Revision is incremented.

Lifecycle Release State is a State within the Lifecycle map that has specifically been flagged as a Released State. There may be more than one Released State in a Lifecycle map, but when an Item that is in the Released State is versioned, its **major_rev** is incremented, it is set to its Lifecycle Start State, and its Revision is incremented.

Promote is an action performed by a user to change the state of an Item. It assigns another State from the same Lifecycle map that is connected by the Transition between the two states.

Edit and Done is a set of actions used to initiate and save changes to Items resulting in Versioning when Automatic Versioning is activated on an ItemType. Edit and done actions explicitly affect the **Generation** and the **major_rev** of an Item.



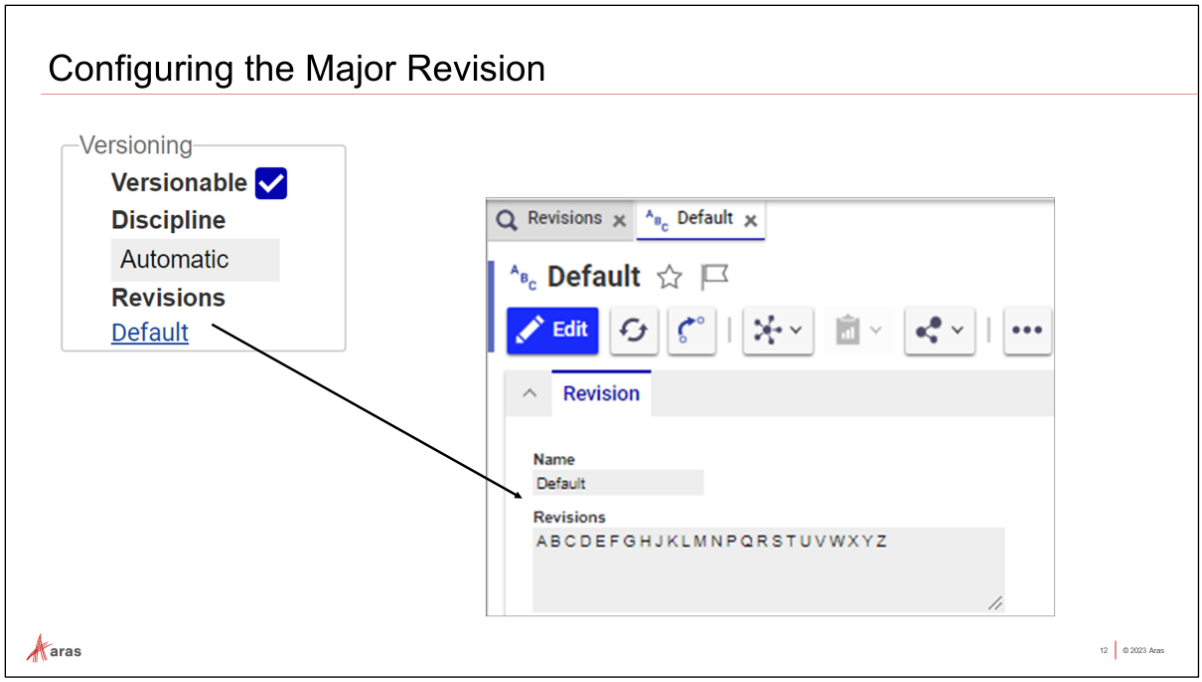
Configuring Item Versioning

Open the desired ItemType and select Versionable as true. The next time an Item is edited and saved after being unclaimed, a new generation of the Item is created using the Automatic discipline.

If the Manual discipline is selected, a user decides when the next generation of the Item is created. The generation is not created automatically.

Try it … Investigate ItemType Versioning

1. Logged in as an Administrator, navigate to Administration > ItemTypes and open the **Part** ItemType for viewing.
2. Notice the Versionable checkbox is checked to enable ItemType versioning.
3. Notice the Discipline is set to Automatic, which increments the generation of an Item.
4. Notice the Revisions is set to Default which dictates how each revision will be labeled.



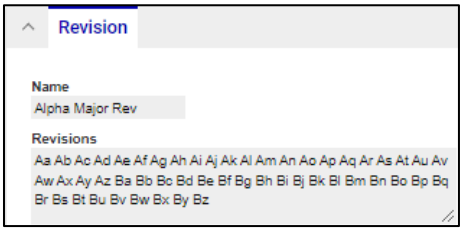
Configuring the Major Revision

Revisions are simply sequences defined in the system as a Revision. The sequence is added to a Revision ItemType separated by a space. The sequence can be any set of numbers and/or characters; it is the whitespace that defines the separation. The sequence is read from left to right top to bottom and can be up to 1024 characters long. It is common to skip revision letters like I and O to avoid confusion with the numbers 1 and 0.

Try it … Create a Major Revision Sequence

1. Navigate to Administration > Revisions and create a new Revision named *Alpha Major Rev*.
2. In the Revisions field, enter in a sequence of letters, such as *Aa Ab Ac Ad Ae Af*, etc., using a space character to separate. Feel free to Copy and Paste this sequence:

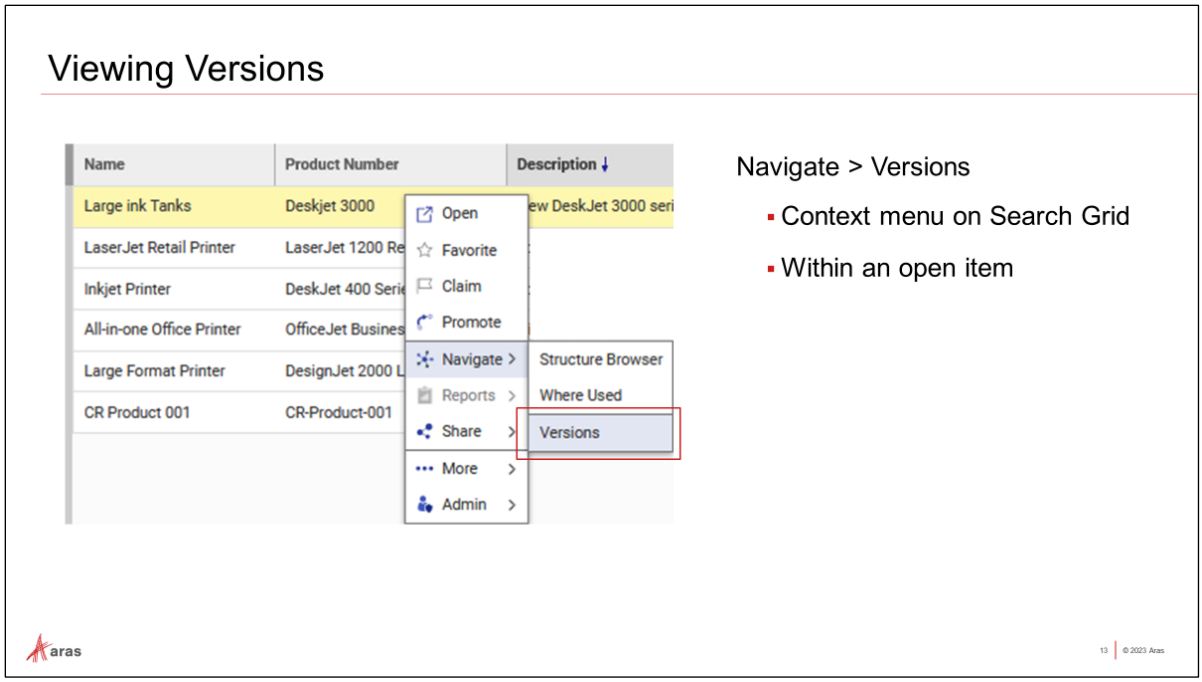
   Aa Ab Ac Ad Ae Af Ag Ah Ai Aj Ak Al Am An Ao Ap Aq Ar As At Au Av Aw Ax Ay Az Ba Bb Bc Bd Be Bf Bg Bh Bi Bj Bk Bl Bm Bn Bo Bp Bq Br Bs Bt Bu Bv Bw Bx By Bz
3. Click Done.



Caution

Create a Revision Item rather than modifying the existing Default Revision. The Default Revision is used internally as you develop a solution and could cause problems with Aras Innovator if modified.

If an item reaches the end of revisions sequence, an error message “failed to version the item: revision sequence exceeded” will pop-up. Analyse your dataset and consider a buffer on the revision sequence.

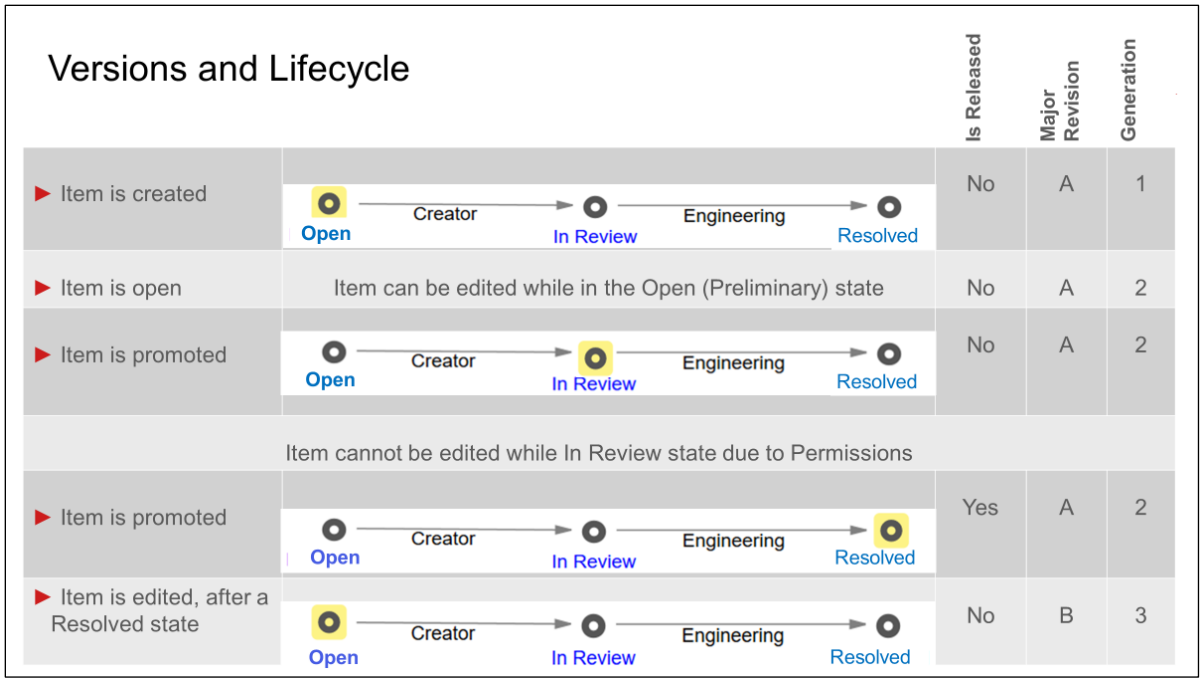


Viewing Versions

Each time a versioned Item is unclaimed, then edited and saved again, a new generation is created. Every generation creates a new row in the ItemType table with a unique Id and the generation property is incremented by one. An Item also maintains a property named config_id. The config_id property contains a 32-character hexadecimal value that remains constant throughout the entire span of an Item's existence. Regardless of how many generations of an Item are created, the config_id is common across all generations. This is how the system maintains traceability of an Item.

Try it … View a Version

1. Navigate to Design > Parts and create a new Part.
2. Type in some property values and click the Done button.
3. Click the Edit button and make a change to one of the property values.
4. Click the Done button once again.
5. Click the Navigate button and select Versions from the menu.
6. In the Item versions dialog window, choose the previous version (generation 1) to view.
7. Right click on the new tab > Dock Tab, to display both versions side by side.



Versions and Lifecycle Maps

When a versionable Item is promoted to a state with the Released Item Setting set to true, two behaviors occur when the item leaves the Released state either by manually revising or creating an ECO.

- The Lifecycle map is reset to the starting State.
- The Major Revision is incremented based on the Revision sequence.

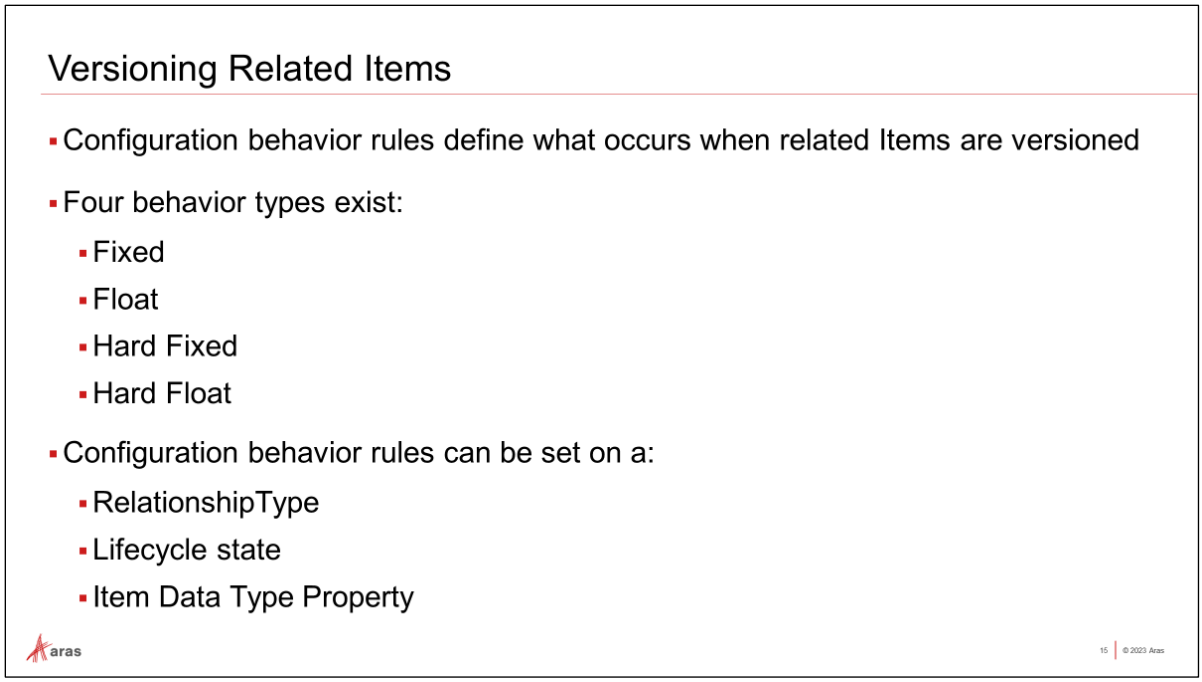
In the simple example above, an Item is created (the Open Lifecycle state) and displays as Revision A – Generation 1 in the Preliminary state.

The Item is then edited (saved and unclaimed) and shows as Revision A – Generation 2 in the same Preliminary state.

Then, the current version of the Item is promoted to the In-Review state. In this example, we have set the state Permissions to prevent editing while an Item is In Review.

Again, the current version of the Item is promoted, now to a state named Resolved.

The next time the Item is edited (Edited, Saved, and Unclaimed) the version is incremented to Revision B, Generation 3, and the current state becomes Open for the new version.



Versioning Related Items

Versioning a single unrelated Item is straight forward, as we have seen. A new generation of the Item is created each time the Item is edited and then saved.

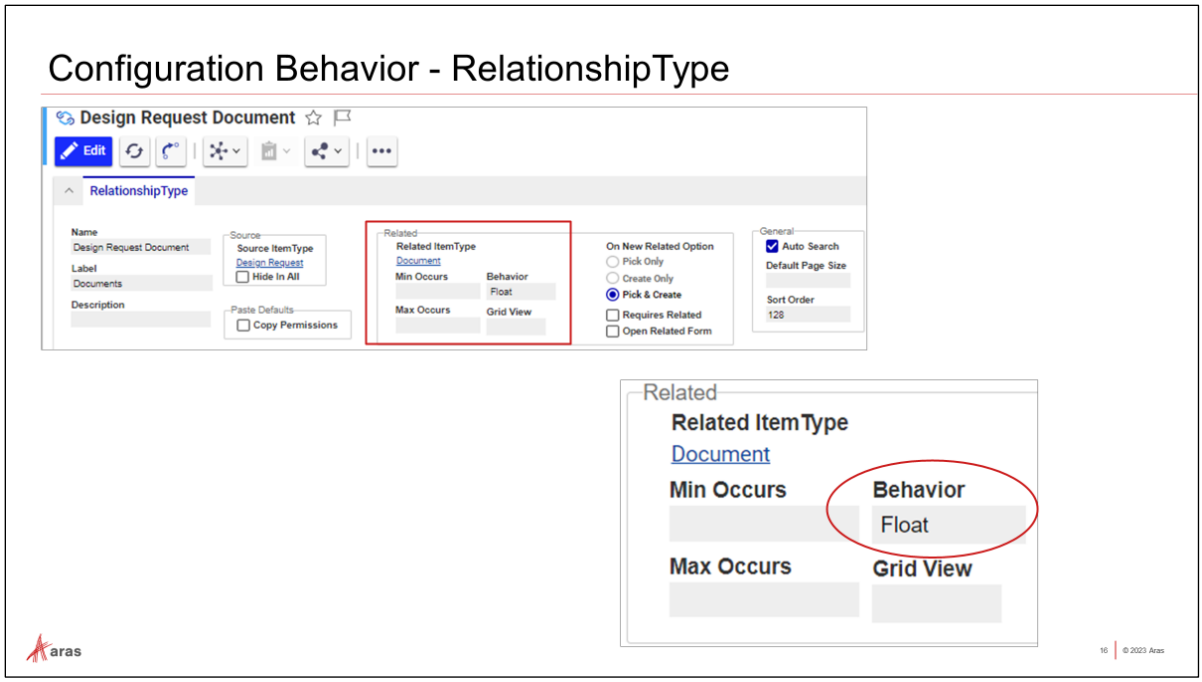
The behavior becomes more complicated when the versioned Item is related to other Items in the database. Should the other Items still relate to the next generation? What if a related Item is edited – which version shall be related to its Parent Item?

Four behavior rules for related Items can be set on a RelationshipType, an Item Property, or a Lifecycle state.

Item Behavior is a property that sets behavior on the connection between source item, or parent, and the related item, or child. Keep in mind that an Item Behavior property is defined on the RelationshipType as well (see RelationshipType Item Behavior). These settings work together to control the configuration of source and related items.

Versioning Behavior rules:

- Fixed: Points to a specific version of a related item, e.g., at the time when it was added to the relationship
- Float: Always points to the current version of the related item
- Hard fixed: Points to a specific version of a related item, this behavior cannot be changed at a later point (e.g., a certain life cycle state)
- Hard float: Points to the current version of a related item, this behavior cannot be changed at a later point (e.g., a certain life cycle state)



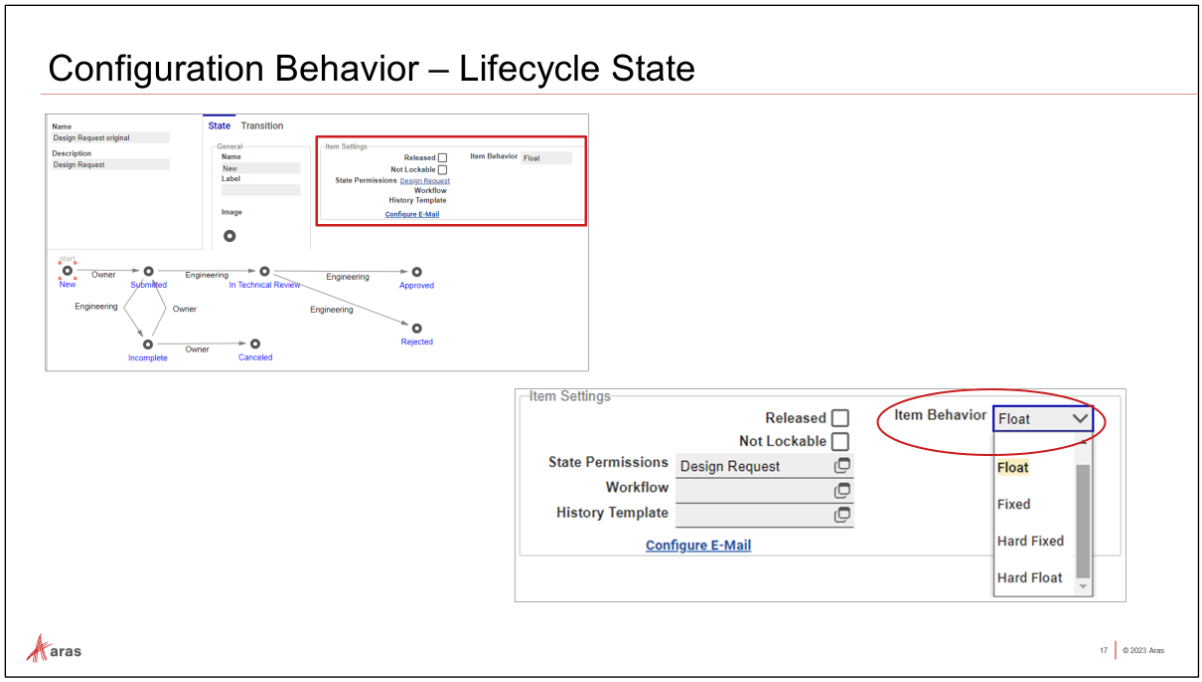
Configuration Behavior - RelationshipType

RelationshipType behavior for related items can be set on a RelationshipType using the Behavior drop-down list.

This rule sets the behavior property on the instance of a Relationship ItemType when it is created.

Try it … Set the Behavior for a Relationship Type

1. Open the **Design Request Document** RelationshipType.
   (Administration > Relationship Types > Design Request Document.)
2. Whenever a Design Request Document is updated, we want the new version of the document to be related to the Design Request; therefore, ensure the Behavior is set to **Float**.
3. If you made edits, click Done.



Configuration Behavior – Lifecycle State

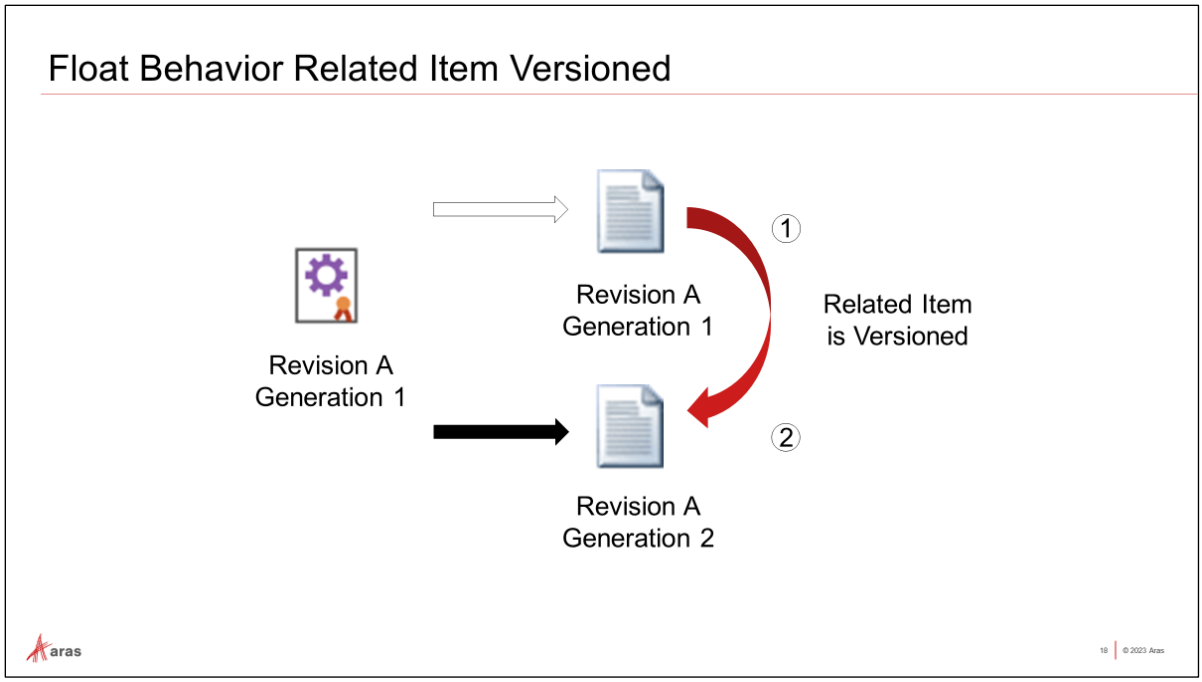
Each Lifecycle state can be configured with an Item Behavior by selecting the behavior from the drop-down list on the state Item Settings.

If Item Behavior is set for a Lifecycle state, it takes precedence over the RelationshipType behavior that may be defined for the Relationship Item.

Try it … Set the Lifecycle State Behavior

We want the behavior for each Lifecycle state to be Float unless the Design Request is being reviewed or is released. Once a DR is being reviewed or is released, the behavior should be Fixed.

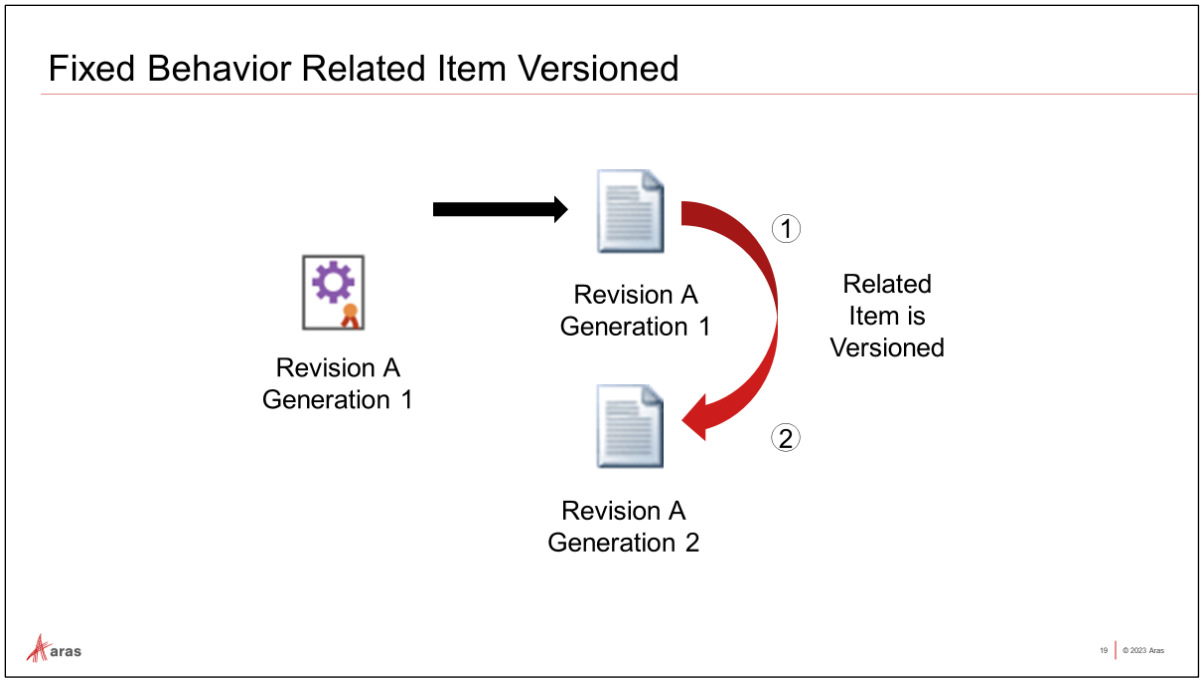
1. Open the **Design Request** Lifecycle map for editing.
2. Set the Behavior to *Float* for the following Lifecycle states: **New**, **Submitted**, and **Incomplete**.
3. Set the Behavior to *Fixed* for the following Lifecycle states: **In Technical Review, Cancelled**, **Approved**, and **Rejected**.
4. Click Done.



Float Behavior Related Item Versioned

In the following case, the source item is not versioned or does not change its generation. If a Lifecycle state behavior is set as Float, when a related Item is revised to a new generation, the source Item points to the newest version of that Item.

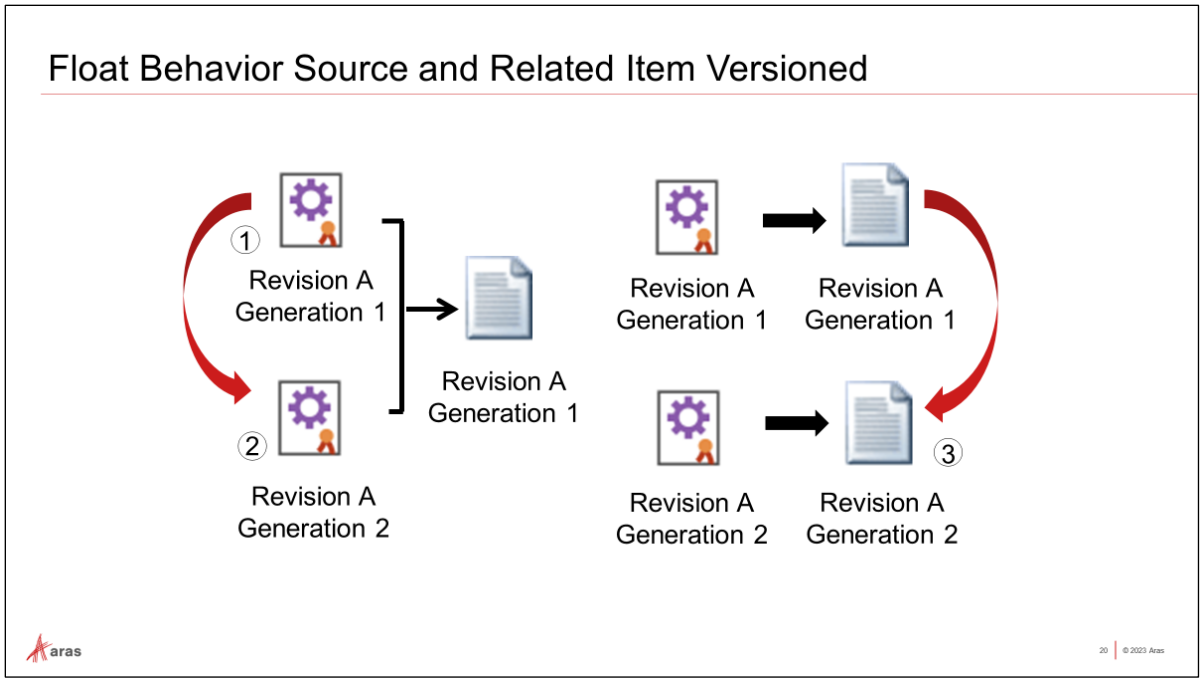
1. In this example, a Design Request was created, and a relationship is established to a Document Item at Revision A – Generation 1.
2. The Document Item was then edited, and a new revision was created (Revision A – Generation 2).
3. The Design Request Document relationship floats to the highest revision of the Document (Revision A – Generation 2).



Fixed Behavior Related Item Versioned

If a Lifecycle state behavior is set as Fixed, when a related Item is revised to a new generation, the source Item continues to point to the existing version of that Item.

1. In this example, a Design Request is created, and a relationship is established to a Document Item at Revision A – Generation 1.
2. The Document Item was then edited, and a new revision was created (Revision A – Generation 2).
3. The Design Request Document relationship remains fixed to the existing Document revision (Revision A – Generation 1).

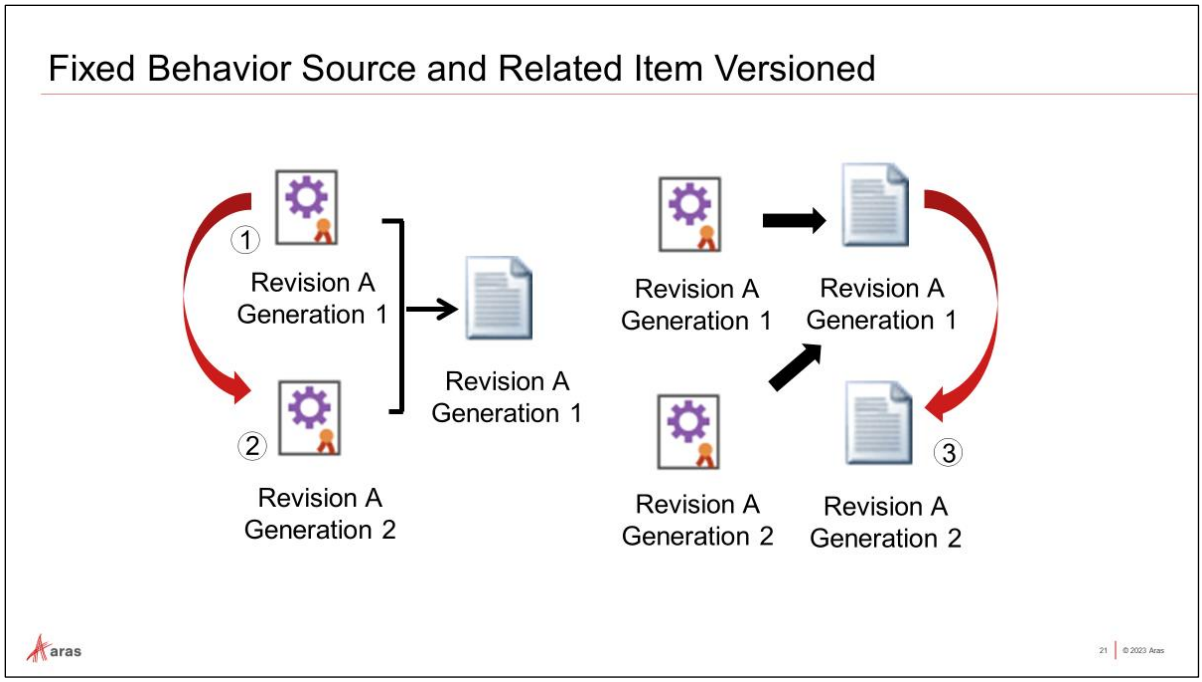


Float Behavior Source and Related Item Versioned

This example demonstrates when both a Source and Related Item are versioned.

1. In this example, a Design Request is created, and a relationship is established to a Document Item at Revision A – Generation 1, and the Lifecycle state behavior is set to Float.
2. The Design Request is then revised (Revision A – Generation 2). Since there are no other Document Items, the relationship to Document A -1 continues.
3. A user then creates a new revision of the Document Item (Revision A – Generation 2). The Design Request (Revision A – Generation 2) now floats to the newer Document Item (Revision A – Generation 2).

Note, however, that the original Design Request (Revision A – Generation 1) remains related to Document Item Revision A – Generation 1. The relationship behavior of Design Request Revision A – Generation 1 to Document Item Revision A – Generation 1 is set to Hard Fixed to maintain this configuration for this version. This behavior is in accordance with the CM2 standard.



Fixed Behavior Source and Related Item Versioned

This example presents the same scenario as the previous page. However, since the Lifecycle state behavior is defined as Fixed, Design Request Revision A – Generation 2 remains Fixed to Document Item Revision A – Generation 1.

1. In this example, a Design Request is created, and a relationship is established to a Document Item at Revision A – Generation 1, and the Lifecycle state behavior is set to Fixed.
2. The Design Request is then revised (Revision A – Generation 2). Since the relationship is Fixed, the relationship to Document Item A -1 continues.
3. A user then creates a new revision of the Document Item (Revision A – Generation 2). The Design Request (Revision A – Generation 2) stays fixed to the original Document Item (Revision A – Generation 1).

Summary

In this unit, you learned how to define a Lifecycle map for an Item as well as version Items in the database. You also learned about related item behaviors when connected Items are revised.

You should now be able to:

- Identify the Lifecycle States
- Define Lifecycle Transitions
- Create a Lifecycle Map for an ItemType
- Define Lifecycle State Permissions
- Configure and View Item Versions
- Define Versioning Behavior

Review Questions

1. What triggers a new Major Revision for an Item?

2. What is the purpose of a Lifecycle transition?

3. Where is Item behavior (e.g., float, fixed) configured as it relates to related Items?

4. What takes precedence?

5. How can I compare two versions of the same Item?

Lab Exercise

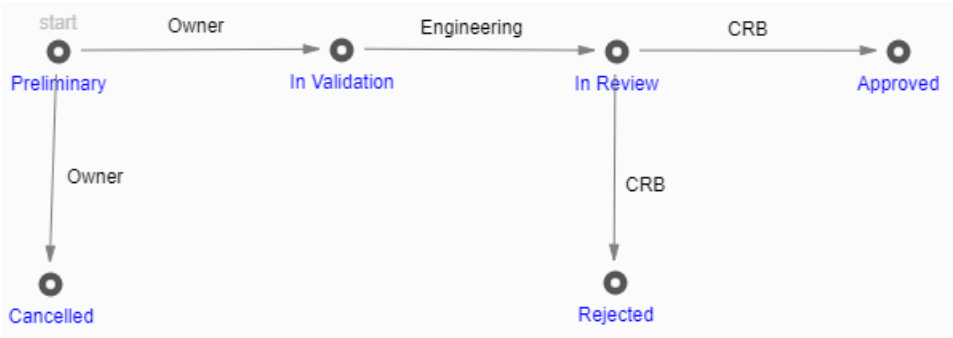
Goal

Be able to define a Lifecycle map for an ItemType and promote an Item through the Lifecycle map. Be able to define relationship behaviors based on the requirements of the situation. Set an appropriate permission for when the request reaches the In-Validation state.

Scenario

In this exercise, you create a Lifecycle map for the Change Request ItemType. You review the relationship behavior between the Change Request Item and related Parts to support a fixed or float versioning scheme.

The following Lifecycle map has been determined for the Change Request:



Defining the Lifecycle

1. Create a new Lifecycle map named *Change Request* and use the graphic above to draw the map.
2. Provide the Roles as shown above for each transition.
3. Provide the following settings for each state:

Lifecycle State	Released	Not Lockable	Item Behavior
Preliminary	No	No	Float
In Validation	No	No	Float
In Review	No	Yes	Fixed
Approved	Yes	Yes	Fixed
Cancelled	No	Yes	Fixed
Rejected	No	Yes	Fixed

4. Click Done for the Change Request Lifecycle map.

   Note that each transition requires an Identity to allow promotion.
5. Open the **Change Request** ItemType for editing and add the new Lifecycle map you created to the Change request ItemType.
6. Use a different browser, login as Diane Prescott (a member of the Engineering identity), and create a new Change Request.
7. Assign the logged in Engineering member as the Owner.
8. Promote the Change Request testing the Lifecycle transitions you created.

   You can view the Lifecycle map from the CR to view the Roles you need for each Promotion (Transition).
9. Logout of the second browser and close it.

Configure a Permission on a Lifecycle State

In the Change Request solution, the Owner can promote a request to the In-Validation state when it is being verified and validated. At In Validation, we want to make sure that both the Engineering and the Change Specialist I role can make edits to the form if necessary while All Employees can only view the form. If an issue arose, the Administrators group should be able to change permissions for a single request at the In-Validation state.

1. Create a new permission named *In Validation Request* and set the permissions as follows:

Name	Get	Update	Delete	Can Discover	Show Permis...	Can change access
Administrators	☑	☐	☐	☑	☐	☑
All Employees	☑	☐	☐	☑	☐	☐
Change Specialist I	☑	☑	☑	☑	☐	☐
Engineering	☑	☑	☑	☑	☐	☐
Owner	☑	☐	☐	☑	☐	☐

2. Click Done, return to the **Change Request** Lifecycle map, and select the **In Validation** state in the map canvas.
3. In the Items Settings area, choose the new **In Validation Request** permission in the State Permissions field.
4. Click Done for the Lifecycle map.

The permission assigned to the Change Request as a default is used when an item is at the Preliminary State. The new permission is used when an item is promoted to In Validation. Since no other Permissions are set for later states, the In Validation permission persists since we do not want edits after Validation.

Working with Versions

1. Open the Part ItemType for editing.
2. Locate the **generation** property in the Properties tab and uncheck (clear) the **Hidden** and **Hidden2** columns and click Done. This allows you to see the generation property in the search grid and relationship grids.
3. Create a new Part Item and set the Assigned Creator and Designated User to **Administrators**.
4. Click Done.
5. Edit the Part and make a small change to the Name property.
6. Click Done.
7. Promote the Part to the **Released** state.
8. Review the versions of the Part by selecting Navigate > Versions.

Defining Relationship Behavior

Next, you observe how relationship behavior can affect the connection to a versioned Item.

1. Create a new **Change Request** Item adding Administrators as the Owner.
2. On the Relationship grid, add a new part using the New Part button.
3. Click Done and note the generation number of the new Part in the relationship grid.
4. Navigate to and open the same new Part for editing and then click Done (to create a new generation of the Part).
5. Now return to the Change Request Item you created previously and notice the new Generation number in the Parts Relationship grid for the part.
6. Promote the Change Request to the Cancelled state.
7. Now create a new generation of the attached new Part outside of the Change Request by navigating back to Design > Parts and make some edits to the part.
8. Click Done and return to the Change Request, refreshing the data.

   Notice the old Generation number in the Parts Relationship grid for the part.

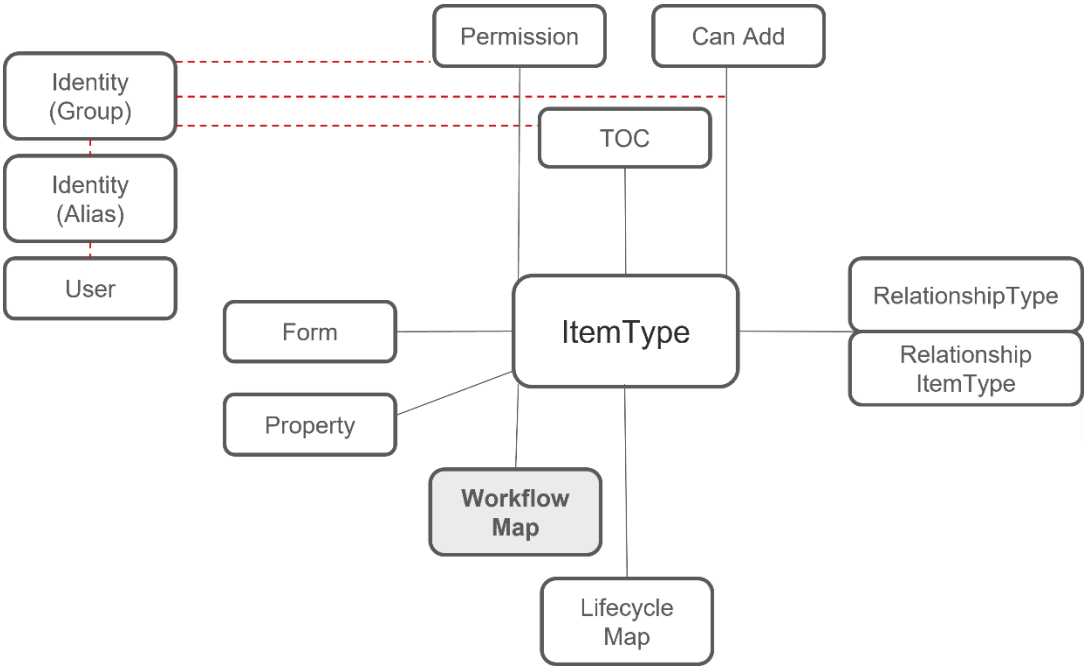
Unit 14 Creating a Workflow Map

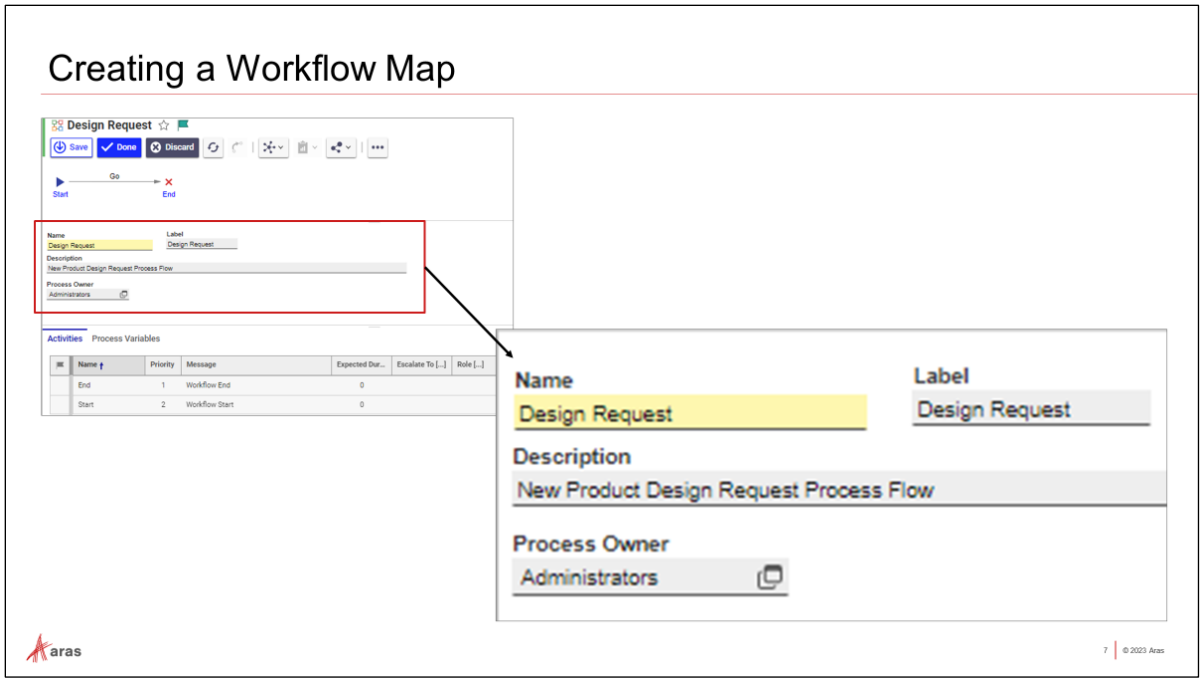
Overview

In this unit, you review a Workflow Map and learn how to create a new Workflow Map that incorporates Activities and Paths. You assign the Workflow to an ItemType and test the new process.

Objectives

- View and create a Workflow Map
- Assign a Workflow Map to an ItemType
- Define an Activity Template
- Create and Configure Workflow Paths
- Assign an Activity to an Identity
- Create a Task
- Define Activity Variables





Creating a Workflow Map

To define a new workflow for an ItemType, you must create a Workflow Map. The Workflow Map determines who receives tasks and assignments and how voting decisions affect which path to take in a process.

Name

Defines the name of the Workflow Map.

Description

Allows you to provide details about this map.

Process Owner

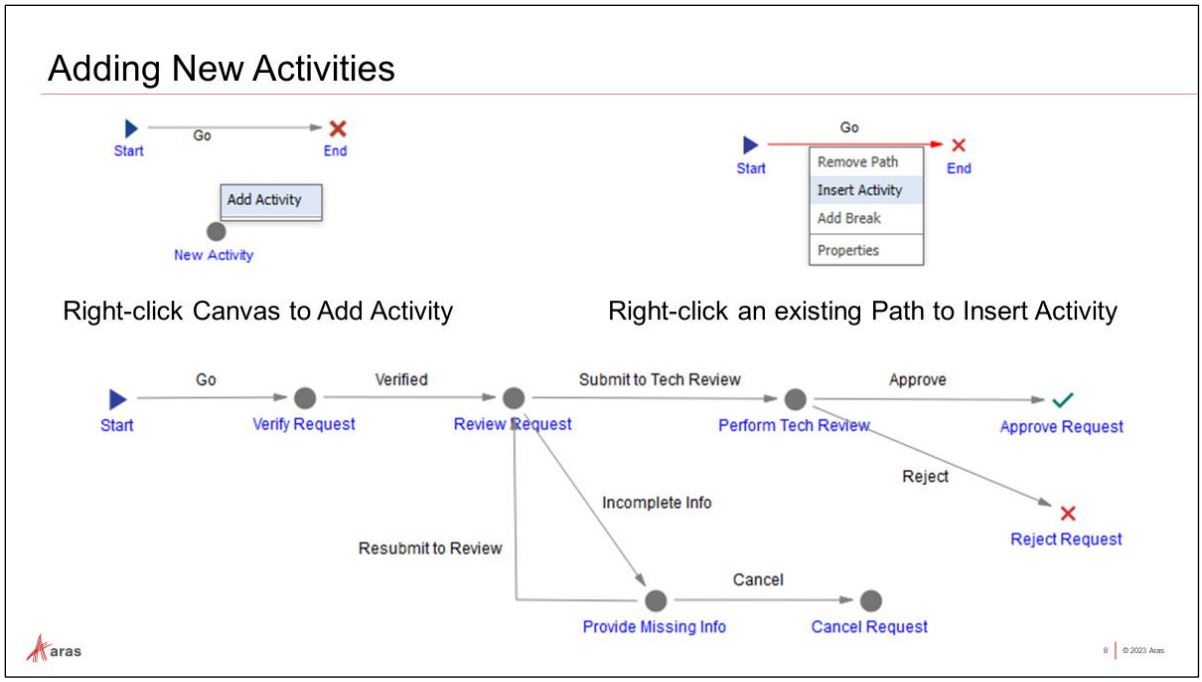
Used to define ownership of the resulting workflow process. The Process Owner is important in escalation strategies and has the authorization to cancel a running workflow.

Try it … Create a Workflow Map

1. Navigate to Administration > Workflow Maps and create a new Workflow Map.
2. Provide the Name, Description, and Process Owner as shown below:

Name	Design Request
Label	Design Request
Description	New Product Design Request Process Flow
Process Owner	Administrators

3. Save the new workflow map.



Adding New Activities

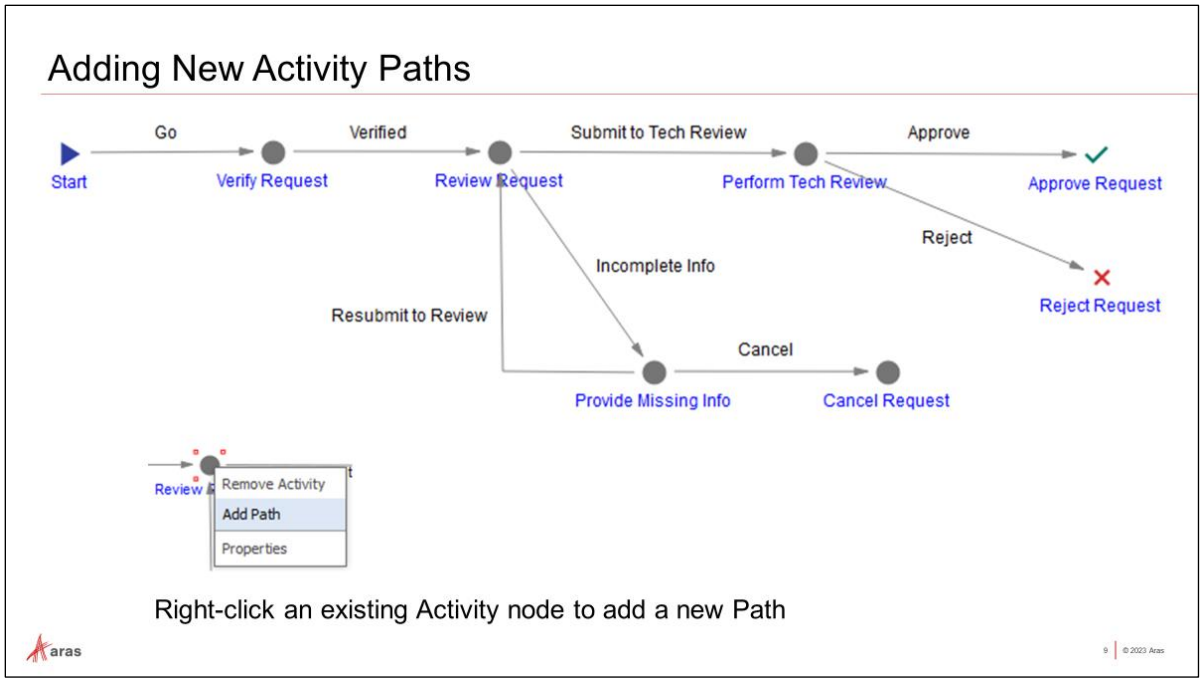
Activities dictate which assignments and tasks are created from a Workflow Process. Think of each Activity as an action someone (or the system) must take to continue the workflow process. To add an activity (node) to the Workflow map, right-click on the Workflow Canvas, select Add Activity from the context menu, and then provide a Name for the Activity in the Activity Template. Activities can also be added by using the context menu on an existing Path.

Try it … Insert New Activities

1. With the Design Request Workflow Map open for editing, insert a new Activity by right-clicking on the **Go** path and choosing Insert Activity from the context menu.
2. Name and label the new Activity node as: *Verify Request*.
3. For the Design Request Workflow Map, add the following additional activities and arrange as shown above.

Review Request
Perform Tech Review
Approve Request
Provide Missing Info
Cancel Request
Reject Request (Rename the End node)

4. Click Save.



Adding New Activity Paths

Workflow paths connect workflow activities to form a process to follow. The Path Label becomes a Voting option for the workflow activity participants. If the Path Label field is not used, then the Path Name displays instead.

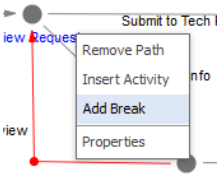
Try it … Add Activity Paths

1. To add an Activity Path, right-click on an Activity and choose Add Path from the context menu.
2. Then, position the mouse pointer on another Activity and click to form the connection.
3. Provide a name for the path in the Workflow Path Template.
4. Add Activity Paths with the suggested settings as shown below:

Go	From Start to Verify Request
Verified	From Verify Request to Review Request
Submit to Tech Review	From Review Request to Perform Tech Review
Approve	From Perform Tech Review to Approve Request
Reject	From Perform Tech Review to Reject Request
Cancel	From Provide Missing Info to Cancel Request
Incomplete info	From Review Request to Provide Missing Info
Resubmit to Review	From Provide Missing Info to Review Request

Adding Breaks in Activity paths

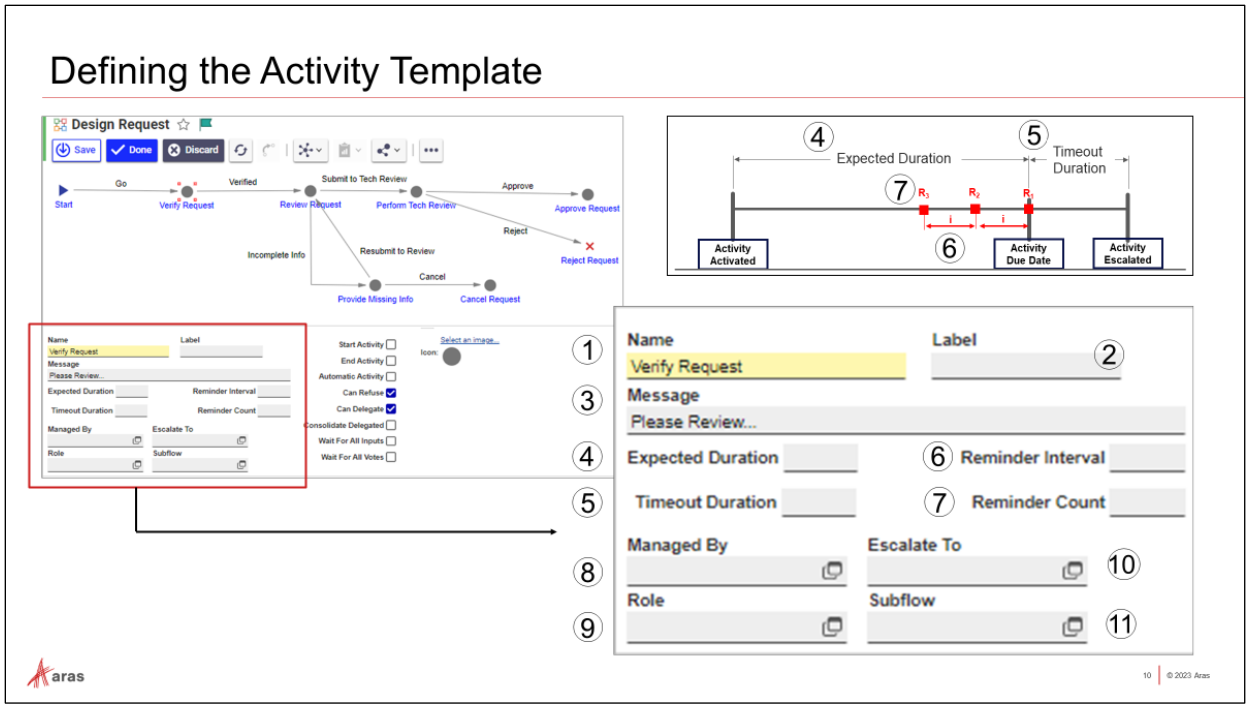
Remember you can right-click an activity path to add a Break point.



Try it … Add a Break

A break allows you to bend an Activity Path to provide a more readable display.

1. Right-click on an Activity Path and select Add Break from the context menu, dragging the break dot to form a bend in the path.
2. For the Design Request Workflow Map, add a break to the **Resubmit to Review** path as shown in the above diagram.
3. You can also move a path name by selecting and dragging it.
4. Save the Workflow.

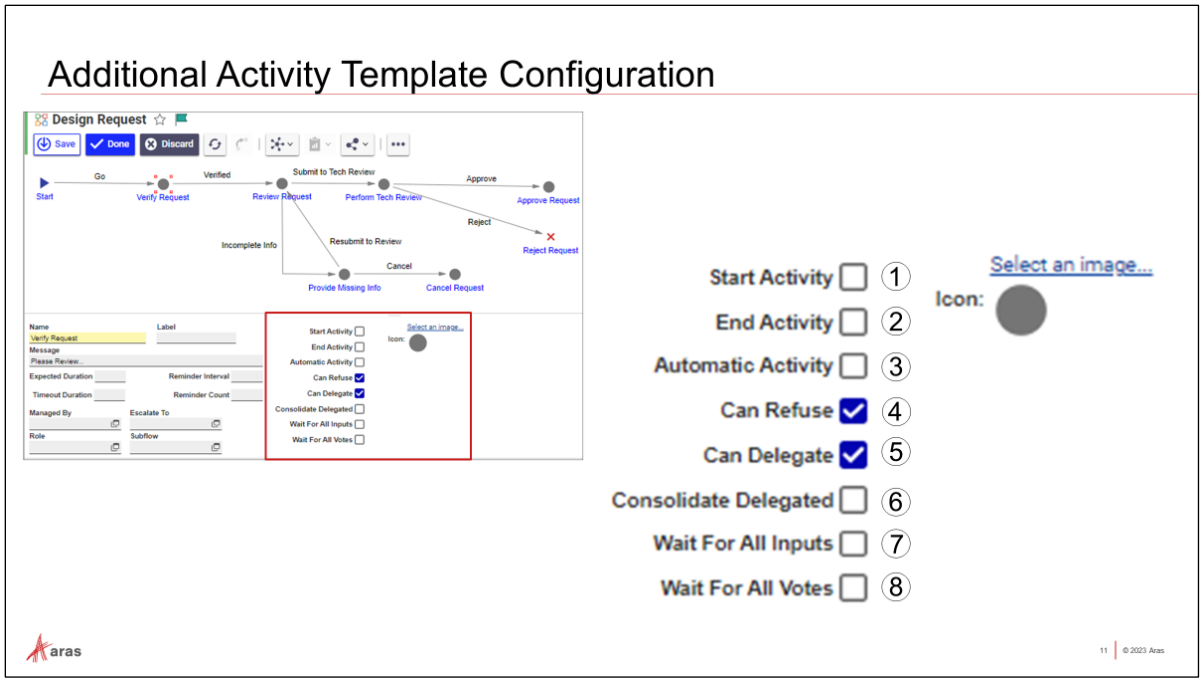


Defining the Activity Template

The following fields are used to define the Activity Template:

①	**Name**	The Name of the Activity.
②	**Label**	The Multilingual label for internationalization.
③	**Message**	The Multilingual message to display to the user in the Assignment.
④	**Expected Duration**	The number of days used to calculate the due date of the activity which is equal to the activity start date plus the expected duration days entered. For example, if the activity starts on June 1 and the expected duration entered is 6, the due date is equal to June 7.
⑤	**Timeout Duration**	The number of days used to trigger escalation events after the due date. If blank, then escalation begins on the due date based on the Expected Duration.
⑥	**Reminder Interval**	The amount of time between reminder messages calculated backward from the calculated due date of the activity. Used in conjunction with the Reminder Count. For example, assume an activity starts on June 1 and is due on June 7 based on an Expected Duration of 6 days. If the Reminder Interval = 2 with a Reminder Count = 2 then a reminder will be sent on June 5 and June 3.
⑦	**Reminder Count**	The number of reminder messages to be sent. The Expected Duration must be greater than Reminder Count * Reminder Interval.

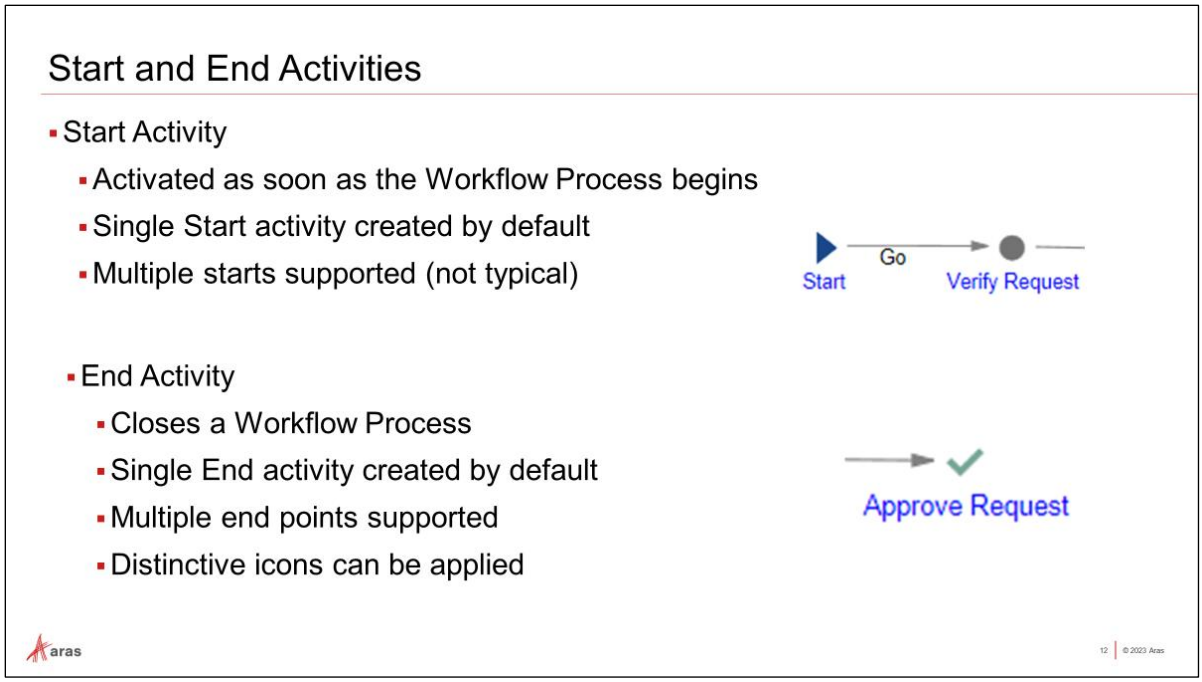
⑧	**Managed By**	An identity that can make changes to the Assignment list of a running activity. For example, you could assign Peter Smith as the Managed By ID of the In-Work activity on the Design Request workflow. This means he could view the running workflow of a Design Request and edit the Assignment(s) on the In-Work activity.
⑨	**Role**	This field is used in conjunction with the Managed By Identity described above. If left blank, the Managed By Identity of an Activity can edit the Assignment list and choose anyone in the system for reassignment. If a role is provided, the user is restricted to reassigning the activity to only members of that role.
⑩	**Escalate To**	The Identity that is notified when an Activity is escalated, i.e., when it becomes overdue, when it is refused, or when no decision was achieved through voting. When there is no identity specified, assignments are escalated to the Process Owner. If no Process Owner Identity is specified, the assignments escalate to the Creator of the Workflow Process.
⑪	**Subflow**	The workflow map to be instantiated as a subflow. An Activity will not be considered complete until its subflow is complete. The exit path from an activity that represents a subflow is typically marked as the Default Path.



Additional Activity Template Configuration

The following additional fields are used to define the Activity Template:

①	**Start Activity**	The first activity to be executed when the workflow begins.
②	**End Activity**	End Activity closes the workflow process. There can be more than one End Activity in a workflow.
②	**Automatic Activity**	This activity has no user interaction and completes automatically.
④	**Can Refuse**	Allows a user to refuse this activity.
⑤	**Can Delegate**	Allows a user to delegate this activity.
⑥	**Consolidate Delegated**	It groups assignments of the same type into a single assignment.
⑦	**Wait for All Inputs**	Does not activate until all input paths connected to this Activity have been completed. If not checked, then the first path that is voted on moves the item through the workflow process. The votes of the remaining paths, if/when voted - are captured but they do not affect the workflow process since the first Vote moved the item through to the next Activity.
⑧	**Wait for All Votes**	Force a delay until all votes are in, even if a path reached is already > 100.



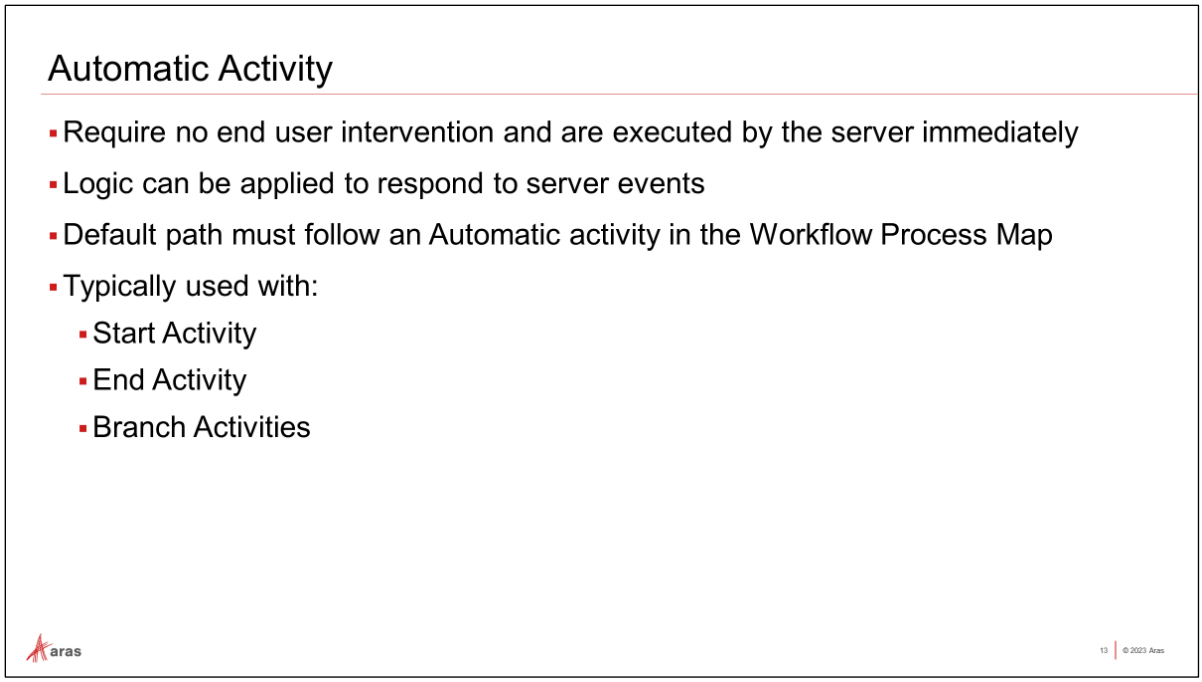
Start and End Activities

It is common to have a single start activity on simple workflows. This supports consistency for automating business processes using a workflow. However, multiple start activities are supported for more complex processes.

Any activities that are set as “End Activity” will cause the workflow process to end whether all assignments have been completed or not. Once a path is voted upon to progress to an “End Activity”, all remaining assignments are removed from the InBaskets of assignees. The Workflow Process Lifecycle state is promoted to a Closed state.

Try it … Set Start and End Activities

1. On the Activity Template for **Start**, ensure that the Start Activity and Automatic Activity are set for it. (They should have been set by default.)
2. Set the three ending nodes (**Approve Request**, **Reject Request** and **Cancel Request**), to an *End Activity*.
3. For the Approve Request activity, click the Select an image… link and use the green checkmark image (Innovator tab > Images folder).
4. Select the Cancel Request activity and change the Icon to the red X image.
5. Save the Workflow.



Automatic Activity

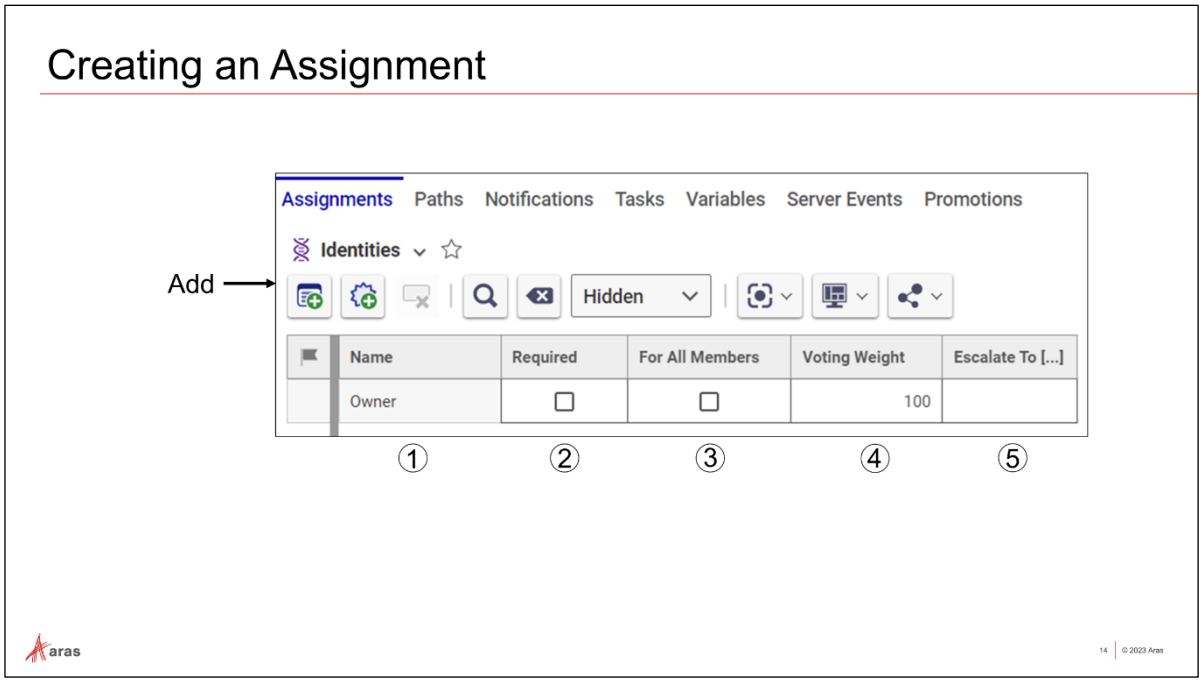
An Automatic Activity requires no user interaction and completes automatically if the Automatic Activity checkbox is selected (true).

- Automatic activities must have a Default Exit Path.
- Automatic activities cannot have any Required Tasks or use Password/E-Signature Authentication.
- The Start Activity is an example of an Automatic Activity.

Try it … Define an Automatic Activity

1. For the Design Request Workflow, set both the **Approve Request**, **Reject Request** and **Cancel Request** activities to *Automatic Activity* (checkbox selected).
2. For the **Approve Request**, **Cancel Request**, and **Reject Request** activities, uncheck the *Can Refuse* and *Can Delegate* checkboxes.

   The following activities should now all be set to Automatic: **Start, Approve Request, Reject Request**, and **Cancel Request**.
3. Save the Workflow.



Creating an Assignment

Assignments are used to define who will perform an operation in a workflow. Assignments can be Voted on, Delegated, Refused, and Automatically Escalated. Assignments display in users InBaskets. The following fields define an assignment:

①	**Name**	The Identity to receive an assignment. Note that aside from any Identity, Team Roles are also supported as Workflow Assignments (Team, Team Manager, Team Member, Team Guest) but the Voting Weight distribution may differ.
②	**Required**	At least one member of a Group identity must complete this assignment (regardless of other voting rights).
③	**For All Members**	This option applies to Team Roles or group identities assigned to this activity; if false, only one member of the group identity must complete and vote on the activity, which then automatically removes this activity from the other group members' InBaskets. If true, all members complete and vote on the activity and it is assigned equal voting weight, with the cumulative weight equal to the assigned voting weight.
④	**Voting Weight**	The percentage of the vote for this assignment. For a path to be selected, it must reach at least a 100% vote.
⑤	**Escalate To**	If the assignment is refused or rejected it goes to this identity. If no identity is specified, then the Activity Template Escalate To identity receives this assignment.

Try it … Create Assignments for all Activities

To create assignments for Activities, click on a Node and locate the Assignments relationship tab. Add a new Assignment relationship by clicking on the Add Identities button.

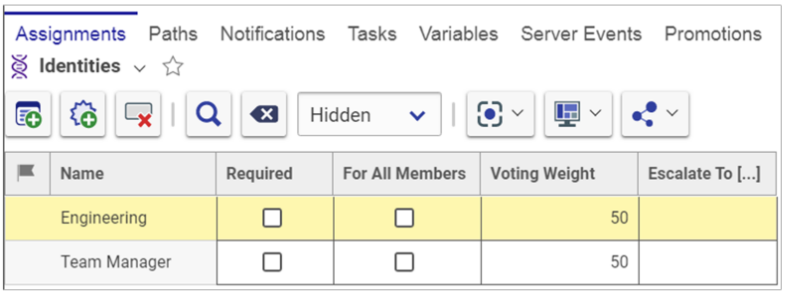
1. For the Design Request Workflow, use the following values to add Assignments to each node:

	Assignments			
Workflow Activity	**Name**	**Required**	**For All Members**	**Voting Weight**
Start	No Assignments, because Start is an Automatic Activity			
Verify Request	Owner	No	No	100
Review Request	Engineering	No	No	100
Perform Tech Review	CRB	No	No	100
Provide Missing Info	Owner	No	No	100
Approve Request	No Assignments, because Approve Request is an Automatic Activity			
Reject Request	No Assignments, because Reject Request is an Automatic Activity			
Cancel Request	No Assignments, because Cancel Request is an Automatic Activity			

2. Save the Design Request Workflow Map.

Defining Voting Weight

- Controls how many assignees must vote a certain way in order to follow a path
- Enables voting rules:
  - Majority
  - Unanimous



aras

15 | © 2023 Aras

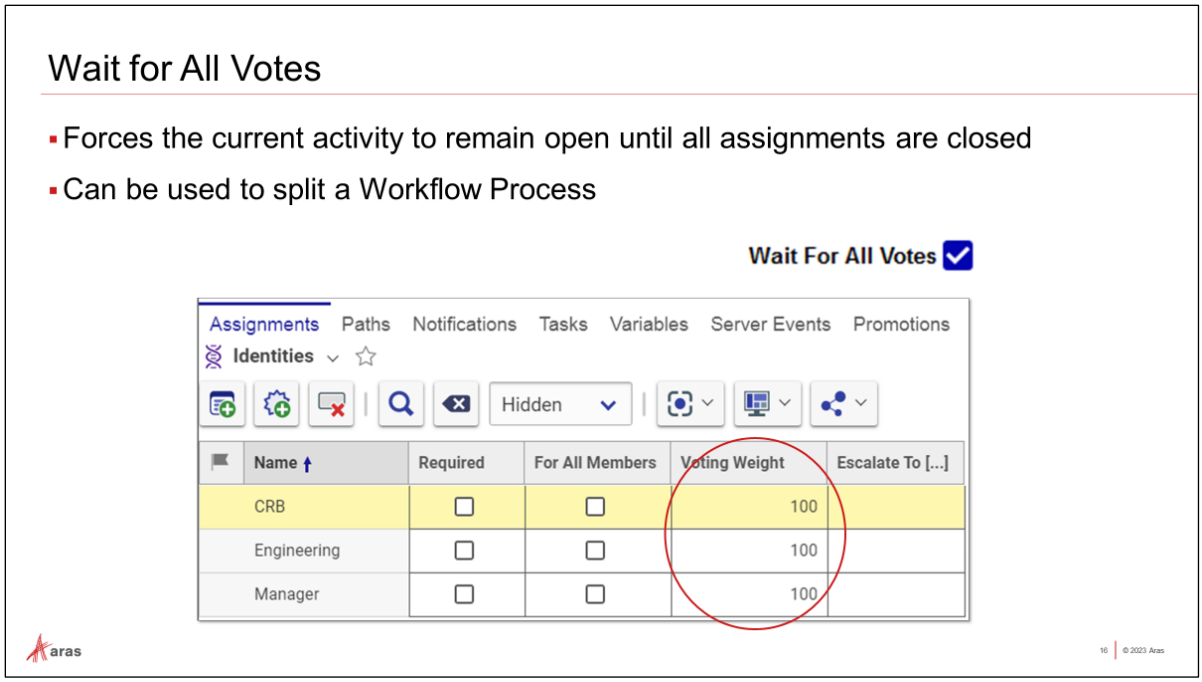
Defining Voting Weight

Each Assignee receives a task in their InBasket for a Workflow Activity and selects an Exit Path using the Assignment Voting Dialog. Each Assignee is given a value or weight for an Exit Path vote, and this is used by the system to determine which path is to be taken. The system Path computation has a combined definition of the Activity and Workflow Map Path.

For path selection to occur, it must reach at least a 100% vote or more. For example (as shown above), assuming no Default and no Override Path is set, the two Identities have 50 as a Voting Weight. Therefore, both identities are required to agree for any Path to add up to 100. Once 100 (or greater) is reached, that Path is followed.

In this example, For All Members is not selected for any group assignment, so only a single member's vote from a group is needed.

If 100 cannot be determined after all votes, then escalation occurs on this activity. This behavior can be adjusted.



Wait For All Votes

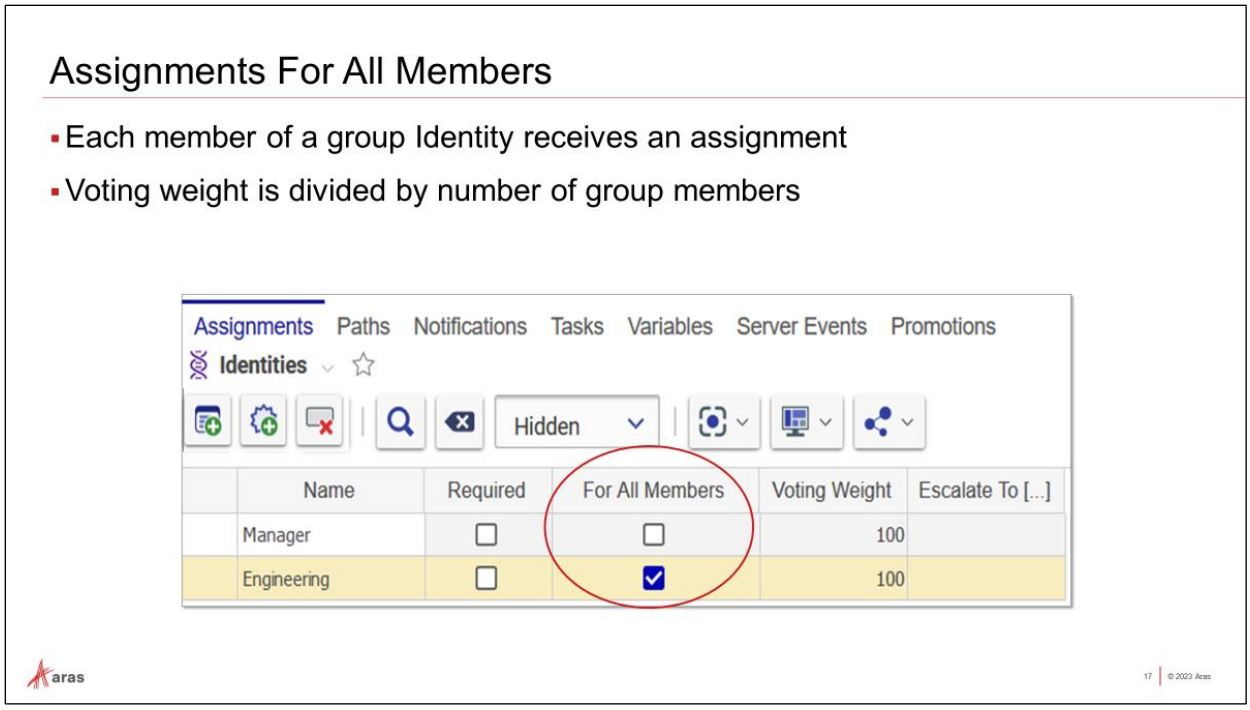
Wait For All Votes does not close an activity until all assignees have voted. Many times, the outcome of a vote is obvious before all votes are in; however, this option forces delay until all assigned votes are gathered.

Alternatively, the Wait For All Votes flag can be used to interactively follow multiple outgoing paths or branches. Exit Paths for an Activity calculate after all votes are gathered. Once gathered, multiple Exit Paths may contain weights of greater than 100. In this case, as with an automatic default path, all Exit Paths with a weight greater than 100 are followed, creating a branch in a Workflow Process.

In the example above, there are three assignees each with a Voting Weight of 100. If the Wait For All Votes flag is not set - an exit path is followed immediately on the first vote since it is >= 100.

Note

Be aware that if the Wait For All Votes flag is selected, accrued weights could be 200 for Exit Path Approve, and 100 for Exit Path Reject. In this case, both the Approve and the Rejected Paths are followed.



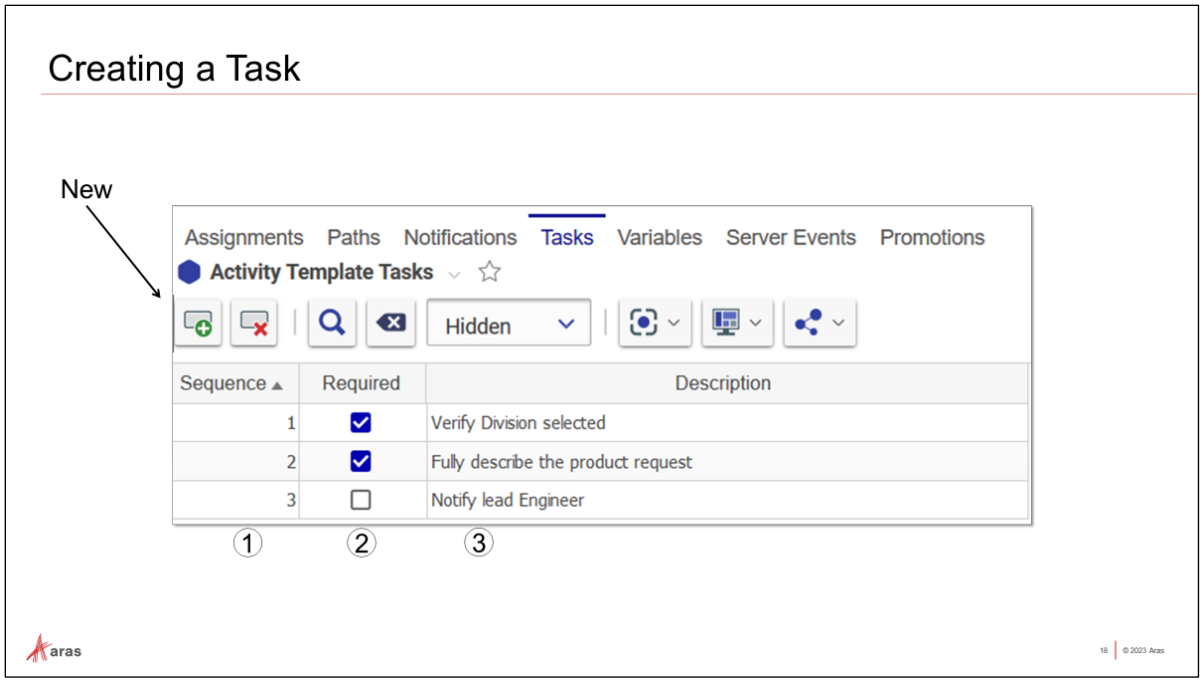
Assignments For All Members

Assignments For All Members applies to Team Roles or Group Identities assigned to an activity. If false, a single member of the Group Identity must complete and vote on an activity, automatically removing this activity from other group members' InBaskets. If true, all members of the Group complete and vote on an activity and are assigned equal voting weight with the cumulative weight of all equal to the Group Assignment Voting Weight.

Voting Weight is calculated at runtime. Notice the Engineering Group has a Voting Weight of 100 and all its members must vote. The Voting Weight is divided by the total number of members in the group. If the Engineering Group had 5 members, each user would have a Voting Weight of 20.

Note

Voting Weight is distributed evenly but it is rounded up. If there are three members of a group identity assigned a weight of 100, each member will get a weight of 34. This may not work as expected when there are more than a dozen members. (For example, 13 and 14 members both result in a weight of 8.)



Creating a Task

Use Tasks to provide some instruction on what work needs to be accomplished to complete an assignment.

The following fields define a task:

① **Sequence**
The order of tasks in the assignment voting dialog.

② **Required**
A task must be completed to complete the assignment.

③ **Description**
Task instructions.

Try it … Create Tasks within a Activity

Locate the Tasks relationship tab for an Activity node and create a new Task relationship.

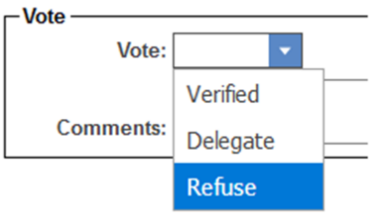
1. For the Design Request Workflow, select the **Review Request** Activity and add the Task values shown below:

Sequence	Required	Description
1	Yes	Verify the Product Division and Product Type selected
2	Yes	Fully describe the product request
3	No	Set Change Type
4	No	Notify the Lead Engineer

2. Save the Workflow.

Refusing an Assignment

- If allowed, assignee can choose Refuse as a vote
- Assignment is then escalated to:
  - Assignment “Escalate To” Identity
  - Activity Template “Escalate To” Identity
  - Workflow Process Owner
  - Creator



aras

19 | © 2023 Aras

Refusing an Assignment

If an Activity allows an Identity to refuse an assignment, an escalation of the assignment begins.

If an Assignment is actively refused, it escalates directly to the Assignment Escalate To Identity. If the Escalate To identity is blank on the Assignment, then the system looks for an Escalate To Identity on the Activity Template. The assignment continues to be escalated, if necessary, to the Workflow Process Owner. If all other Escalate To Identities are blank, the system notifies the Creator of the Workflow Map.

Delegating an Assignment

- If allowed, assignee can delegate assignment to another identity
- If the Activity supports Consolidated Delegation, multiple delegations of the same assignment become one
  - Voting weight is cumulative
  - Reduces the number of assignments in an InBasket

aras

20 | © 2023 Aras

Delegating an Assignment

A user can delegate an assignment to another valid Identity in the system.

Consolidate Delegated

The Consolidate Delegated flag is useful when one activity is delegated to the same Identity several times. Normally, the targeted user would receive multiple assignments to perform the same operation.

For example, an activity has three Assignees each with a Voting Weight of 40. When the Activity becomes active, each Assignee gets a task in their InBasket. Assignee One and Assignee Two both delegate to Assignee Three.

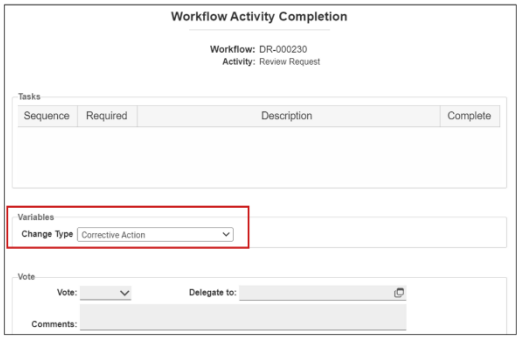
For this example, if the Consolidate Delegated flag is NOT checked, Assignee Three receives three identical assignments - each with a Voting Weight of 40. One is their own assignment and the other two were delegated from Assignees One and Two. Assignee Three must now open each assignment individually and vote a total of three times for the system to calculate a Path.

If the Consolidate Delegated flag is used, Assignee Three sees only one assignment in their InBasket with a Voting Weight of 120 and only needs to vote once.

Another benefit of the Consolidate Delegated feature is in the timing of the vote. If Assignee Three in the above example already voted – when Assignee One and Assignee Two delegate their votes later - the system automatically consolidates their votes (and their weight). Assignee Three does not have to process any further assignments.

Defining Activity Variables

- Store additional information about the Activity on runtime
- Variables not marked as Hidden are available for data entry in the Voting Dialog



Sequence	Name	Type	Source [...]	Default Value	Required	Hidden	Label
1	Select Change Type	List	Change Types		☐	☐	Change Type
①	②	③	④	⑤	⑥	⑦	⑧

aras

22 | © 2023 Aras

Defining Activity Variables

Use the Variables tab to define additional variables that prompt the user for more information in the voting dialog and are captured in the database. The following fields define an Activity Variable:

①	**Sequence**	The Order of variable prompt in the voting dialog.
②	**Name**	The Name of the variable.
③	**Type**	The Data type of the variable.
④	**Source**	The Data Source of the variable (if reference data type).
⑤	**Default Value**	The preset value to show to the user.
⑥	**Required**	The voting dialog cannot be completed unless a value is entered for this variable.
⑦	**Hidden**	Do not show the variable (for programmatic use only).
⑧	**Label**	The Multilingual string for internationalization.

Activity Variables

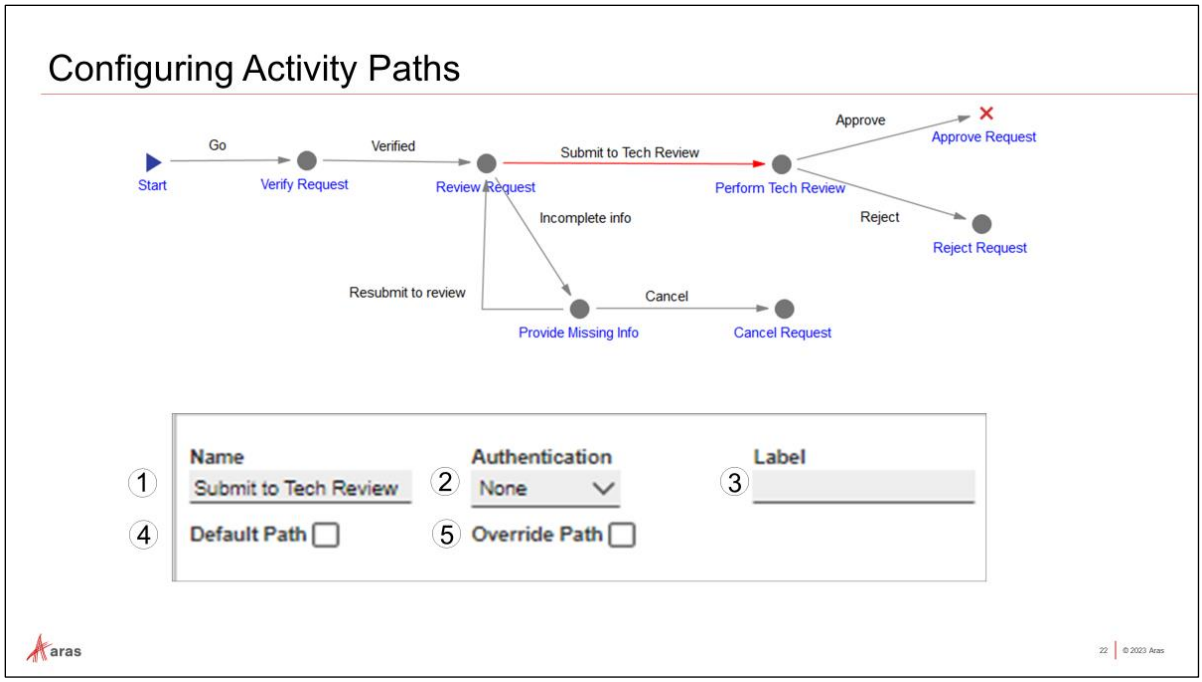
Developers can use variable values collected during voting to programmatically steer workflows down certain paths or to call subflows.

Try it … Create an Activity Variable

1. Create an Activity Variable by selecting the **Review Request** Activity, opening the Variables relationship tab, and clicking the New Activity Template Variable button.
2. For the **Review Request** Activity, add the following variable:

Activity	**Variable**	
Review Request	Sequence	1
	Name	Select Change Type
	Type	List
	Source	Change Types
	Default Value	None
	Required	No
	Hidden	No
	Label	Change Type

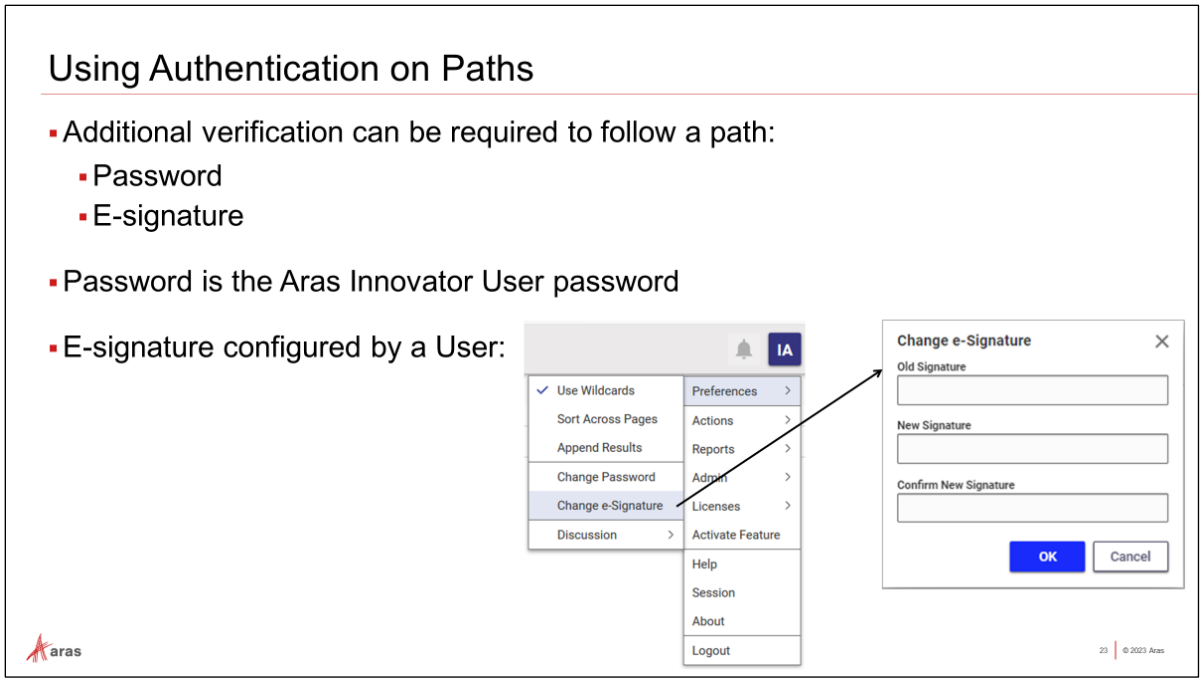
3. Save the Workflow.



Configuring Activity Paths

Use Activity paths to indicate how each Activity interacts with each other in a workflow.

①	**Name**	The Name of the Path.
②	**Authentication**	Require a Password/E-Signature – does the user require authentication to move forward in the process?
③	**Label**	The Multilingual string for internationalization.
④	**Default Path**	This path will be followed if no cumulative voting weight reaches 100 for a single path.
⑤	**Override Path**	If this path is chosen when voting on an activity, no additional votes are collected and this path is followed.



Using Authentication on Paths

If Authentication is selected on an Activity Path, the process user will need to provide credentials to the system to move forward.

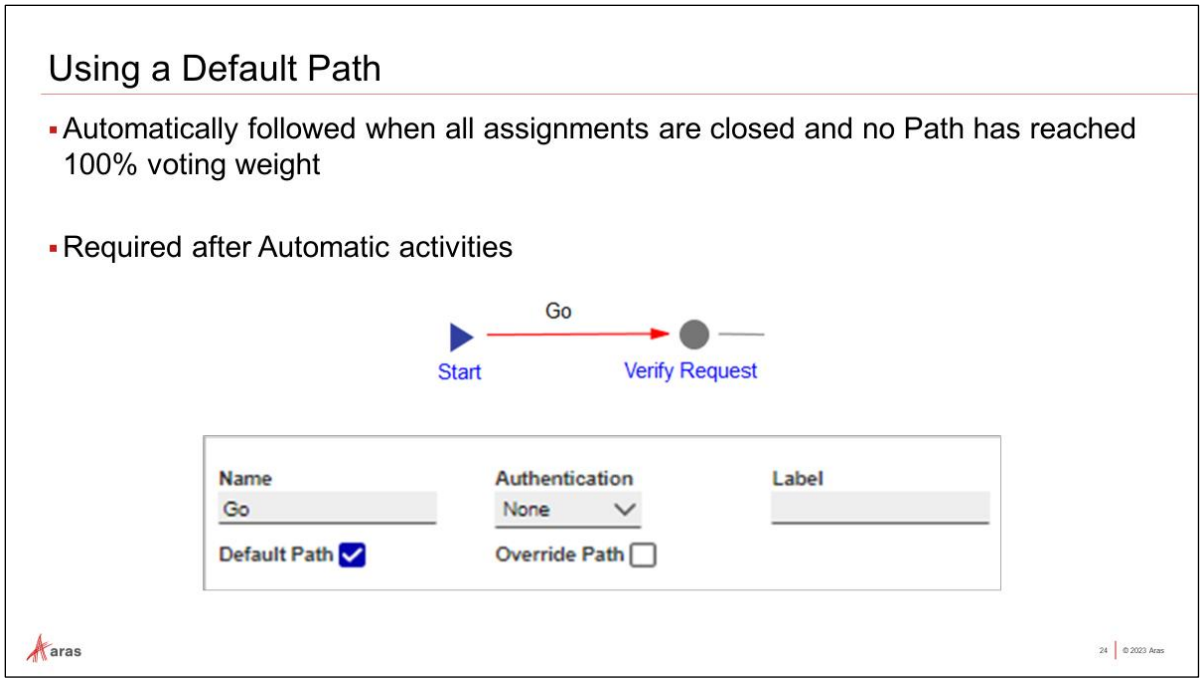
Authentication requires the user to enter a Password or an e-Signature. A vote is not completed until validation is established.

If Password Authentication is selected for a path, end users will need to provide their Aras Innovator password to process the assignment.

An e-Signature is separate from a password. E-signature is a user-controlled password maintained by the user, not an Aras Innovator Administrator or IT password. This allows complete control over an e-Signature by each User. Aras Innovator E-Signatures pass requirements for Military Standards and the Medical Device Industry (21 CFR Part 11).

E-Signatures

- Users manage their own e-Signature from the User Menu by selecting Preferences > Change E-Signature.
- To change an e-Signature, the user provides the current signature (if it exists) and a new signature.
- If a user is setting the e-Signature for the first time, leave the old signature blank.



Using a Default Path

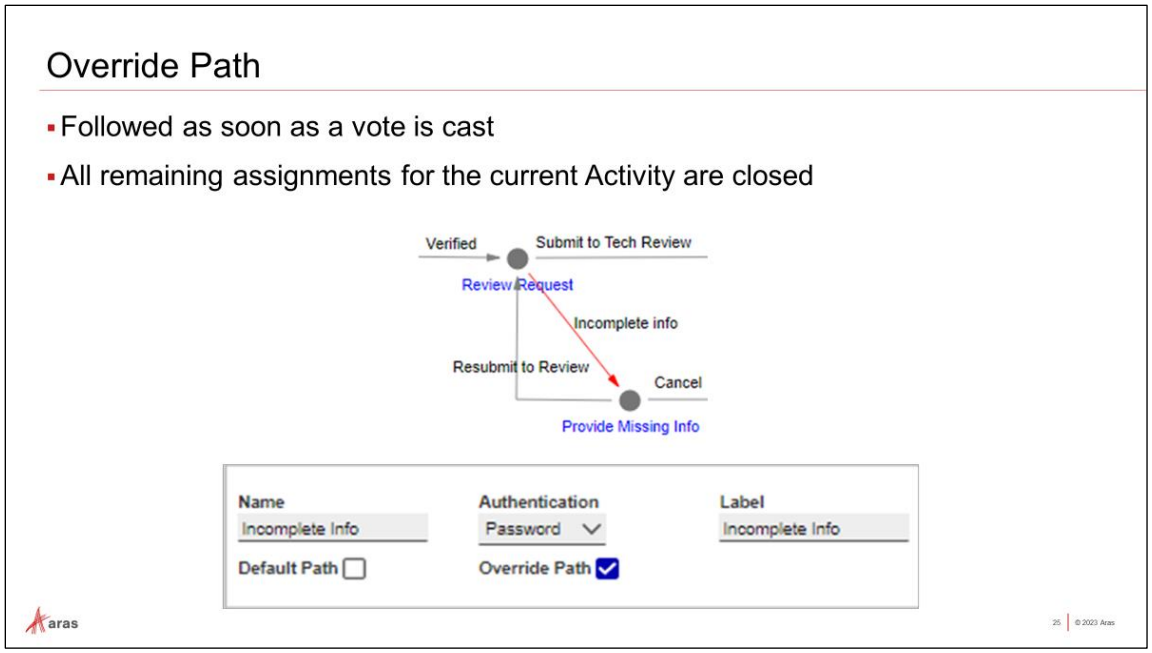
The Default Path setting serves two purposes in a workflow.

The primary purpose is to define a Path for a workflow to follow when no cumulative Voting Weight reaches 100 or more for a single Path from an Activity. Normally, the process will escalate (using the Escalate To Identity set in the Activity Template) if there is no majority decision. To avoid escalation, you can define one or more of the Paths following an Activity as the Default Path. Once all votes have been tallied, if any one path does not have a weight of 100 or more, it can be designated as the default path. If more than one path is marked as Default, then the workflow follows the parallel paths.

The other purpose of Default paths is to indicate an exit path for Automatic activities. At least one Path following an Automatic Activity must be defined as the Default path. If multiple paths are marked as Default, all default paths are followed. Multiple default paths are often used with an Automatic Activity to support branching (parallel processing) in a workflow.

Try it … Set Default Path after Automatic Activity

1. For the Design Request Workflow map, select the **Go** Activity Path that follows the Start Activity, and verify that the Default Path is checked.
2. Save the Workflow.



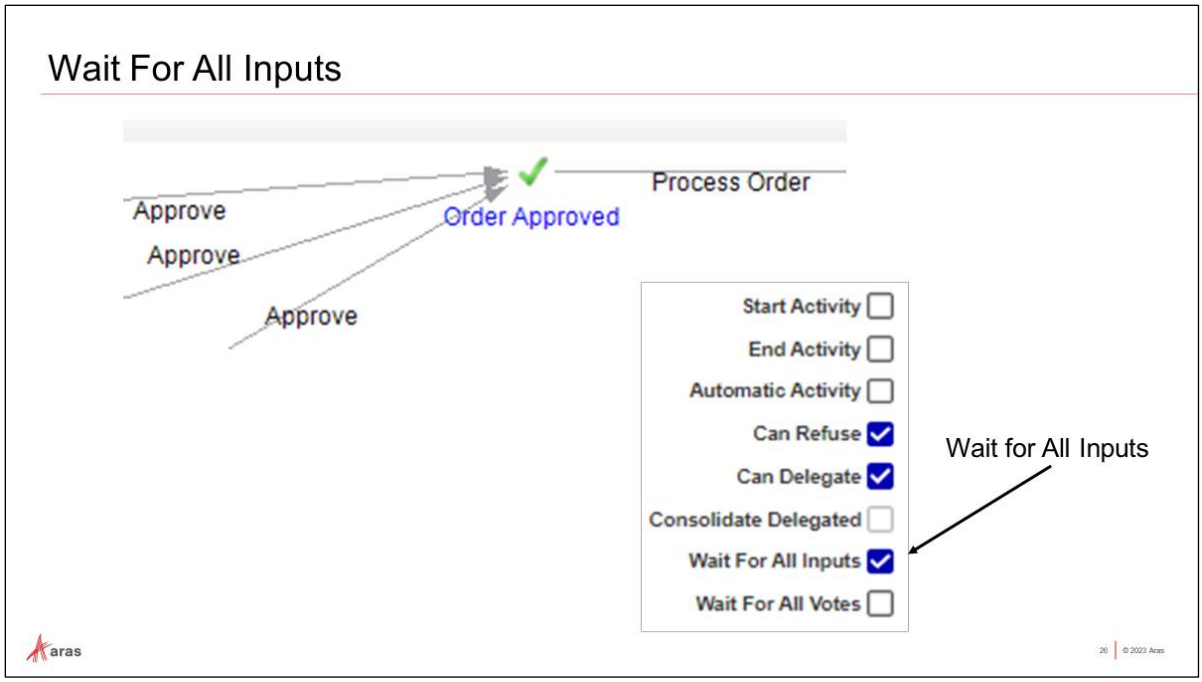
Override Path

Use Override Paths to potentially make a workflow more efficient. If a user chooses an Override path, no additional votes are collected for the activity, and processing continues using the Override path. Override paths take precedence over the Wait For All Votes option.

In the example above, the Incomplete Info path is configured as an Override Path. If any assignees choose this path, any votes for Submit to Tech Review are ignored and the Provide Missing Info activity is invoked.

Try it … Set an Override Path with Authentication

1. For the Design Request Workflow map, on the **Incomplete Info** path, select the Override Path checkbox.
2. For the Authentication field, select **Password**.
3. Click Done.

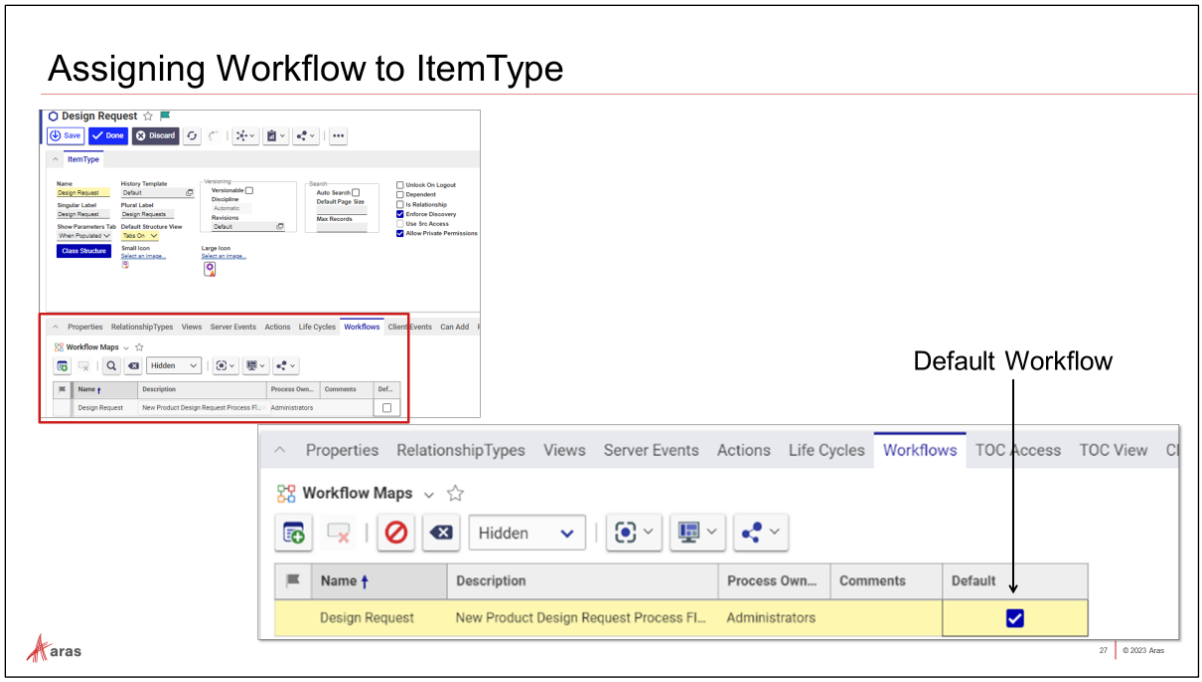


Wait for All Inputs

Wait For All Inputs is used to control when the workflow will activate an activity if two or more paths are connected on the inbound side of the activity.

In the example above, the Order Approved activity has Wait For All Inputs selected. The activity will not be reached until all the Approve paths are processed.

Conversely, if Wait For All Inputs is deselected, then the first path that is processed activates the next Activity (Order Approved in this example) and moves forward. The votes of the remaining paths sit in their respective InBaskets, and the votes of the remaining paths are captured (if voted) but they do not affect the workflow process since the first Vote moved the item through to the next Activity.



Assigning Workflow to ItemType

Before a workflow can be executed as an Item process, it must be associated with an ItemType. An ItemType can be linked to more than one Workflow Map. If the default box is checked, a Workflow Process item is automatically instantiated when creating a new item of the corresponding ItemType.

When an ItemType has a LifeCycle state:

- You should consider instantiating the Workflow from a LifeCycle state.
- Also, Workflows are intended to work with non-versionable Items. Remember that each controlled Item is associated with a Workflow Process Item. Supporting multiple versions of Items with attached Workflow Processes can cause unexpected results.

Try it … Assign a workflow to an ItemType

1. Navigate to Administration > ItemTypes and open the **Design Request** ItemType for editing.
2. Locate the Workflows relationship tab, click the Add Workflow Maps button, and add the **Design Request** Workflow map.
3. Set the Default option to true.
4. Click Done to save and establish the relationship.

Try it … Test the Workflow

1. Signed in as an Administrator (your student login), test the workflow by creating a new **Design Request** Item.
2. Assign Administrators as the Owner.
3. Complete the assignments through to Approved for the Design Request using My InBasket.
   - As the Owner, Verify the Design Request.
   - As an Engineering identity (Diane Prescott), complete the required tasks, select a Change Type, and vote Incomplete Info.
   - As the Owner, vote to Resubmit to Review.
   - As an Engineering identity, complete the required tasks, select a Change Type, and vote Submit to Tech Review.
   - As a member of the CRB (Diane Prescott), perform the Tech Review and vote to Approve.

Summary

In this unit, you learned how to create and configure a new Map.

You should now be able to:

- View and create a Workflow Map
- Assign a Workflow Map to an ItemType
- Define an Activity Template
- Create and Configure Activity Paths
- Assign an Activity to an Identity
- Create a Task
- Define Activity Variables

Review Questions

1. What is the difference between a Workflow Map and a Workflow Process?

2. How do I configure an assignment so that all users belonging to a Group Identity must complete and vote on the activity?

3. How can you create a parallel branch in a Workflow?

4. What must follow an automatic Activity?

5. What two forms of authentication are available to restrict who can complete an assignment?

Lab Exercise

Goal

Be able to define a new Workflow Map and configure activities with assignments and tasks. Be able to create transition paths between activities.

Scenario

In this exercise, you create the basic Workflow Map for the Change Request Workflow.

Requirements

The following requirements have been established for the first phase of the Change Request Workflow. Draw the workflow below before attempting to create it as a Map.

1. Once a user creates a Change Request and assigns an Owner, the owner should receive an InBasket assignment to recheck the request before submitting it for verification/validation and review. If the Owner cancels the request the process should end. If the Owner attempts to cancel the order, the Owner must provide their e-Signature password.
2. If the Owner submits the request to verification/validation, it should be sent to the Engineering group so that any member can verify the request. At the same time, a member of the Change Specialist I Identity should receive an assignment to validate the request.
3. Once both the verification and validation activities are completed, the request should then be submitted to the CRB group for the final review.
4. A member of the CRB must either Approve or Reject the request which ends the process.

Draw the Workflow Diagram:

Steps

1. Create a new Workflow Map named *Change Request* using the diagram you drew above as a guideline. You will need to create new Activities and connect them with Transition Paths. Name the Activities and Paths using the suggested names in the following table.
2. Set the Process Owner as *Administrators*.
3. Define the Assignments as follows:

Activity Name	Assignment Identity	End Activity	Automatic Activity	Tasks
Confirm Request	*Owner*	*no*	*No*	*None*
Cancel Request		yes	yes	
Branch		*no*	yes	
Verify Request	*Engineering*	*no*	*no*	*Review reason and ensure all fields completed*
Validate Request	*Change Specialist I*	*no*	*no*	*Validate cost estimate.*
Review Request	*CRB*	*no*	*no*	*Enter any recommendations in the provided field.*
Approve Request		yes	yes	
Reject Request		yes	yes	

4. Apply the settings so that the Branch activity should split when the reaches it and then process both the Verify Request and Validate Request activities in parallel. For a hint, look at the Answers Key appendix.
5. The path from Confirm Request to Cancel Request should require an e-Signature.
6. Apply the settings so the Review Request Activity assignment is not reached until the Verify Request and Validate Request activities have been completed. For a hint, look at the Answers Key appendix.
7. None of the Activities should allow Refusal except for the Review Request. If the Review Request is refused, it should be escalated to the Administrators Identity.
8. Click Done for the Workflow Map and navigate to Administration > ItemTypes, add the Workflow Map to the Change Request ItemType as the Default, and click Done.
9. Create a new Change Request, assigning the Engineering group identity as the Owner and click Done.
10. Log on as an Engineering group member and view the InBasket.
11. Vote to Submit to Verify the Item.
12. Log on as the member for the next assignment and perform the task to complete the workflow process.
13. Walkthrough the complete workflow process noting the Identities that are assigned to each activity.

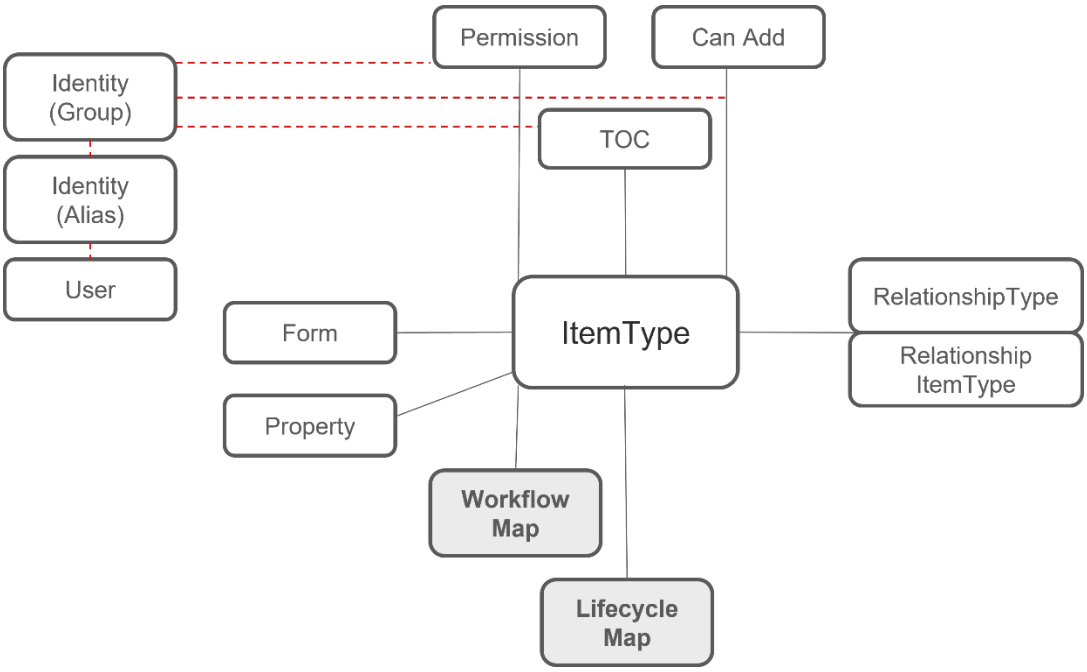
Unit 15 Enhancing a Workflow

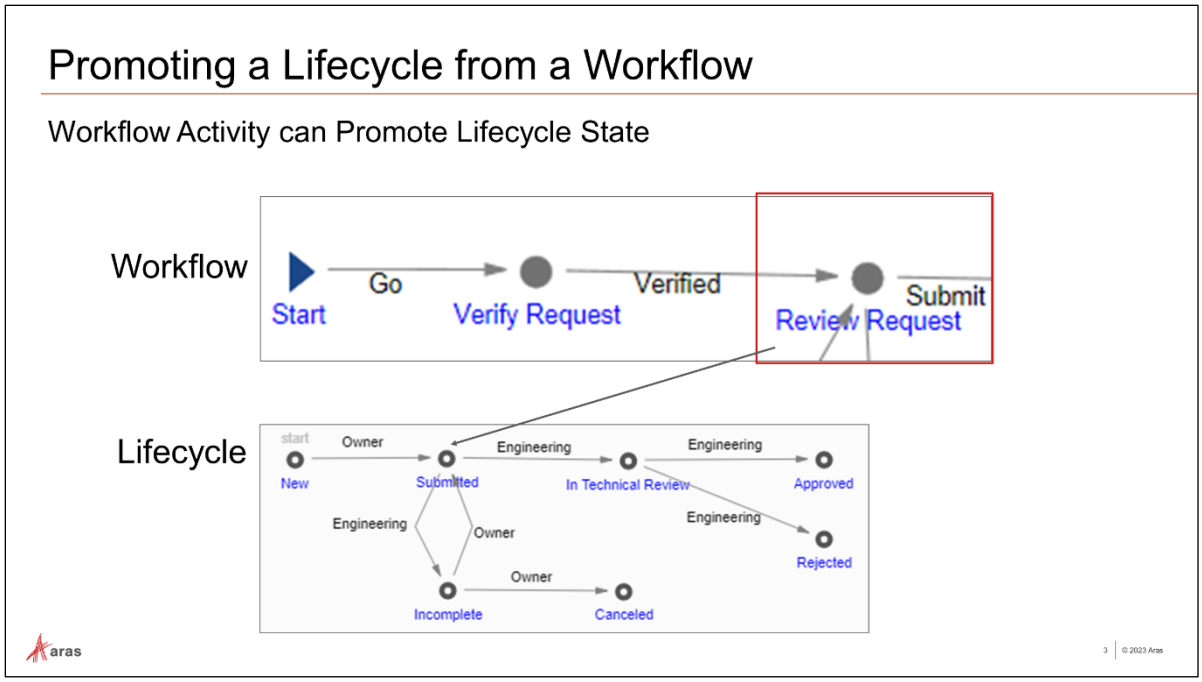
Overview

In this unit you will learn how to integrate an Item Lifecycle with a Workflow. You will also learn about Subflows and Dynamic Assignments.

Objectives

- Promote an Item from a Workflow
- Launch Workflows from a Lifecycle State
- Create and call a Subflow
- Change Assignments on Active Workflow Processes



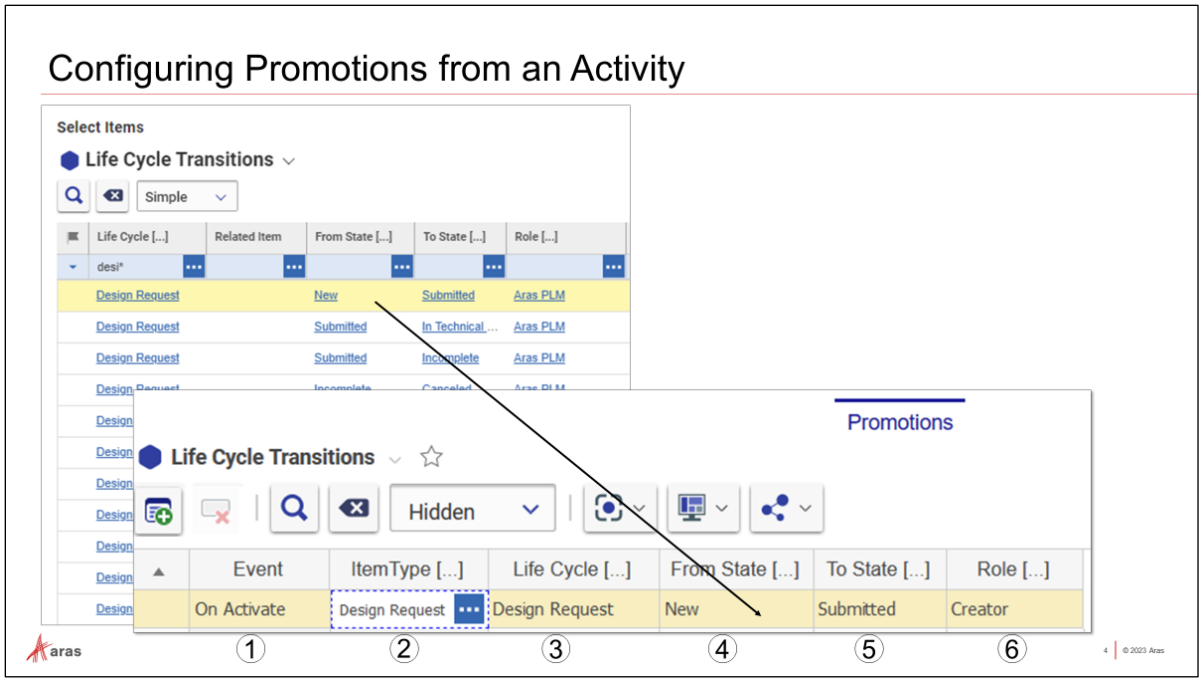


Promoting a Lifecycle State from a Workflow

A Workflow can be configured to provide automatic promotion of the Controlled Item when an Activity is reached in a Workflow process, using the Aras PLM identity.

In the example above, when the Workflow reaches the Review Request Activity, the Lifecycle State of the Controlled Item is automatically promoted from New to Submitted.

Using a combination of Workflow maps and Lifecycle maps provides the best of both technologies. A Controlled Item gains all the advantages of Lifecycle States you have learned about in this course. The Workflow can create assignments, tasks, and use voting weight to decide when an object should be promoted to the next Lifecycle State.



Configuring Promotions from an Activity

Aras Innovator provides the ability to create Workflows that automatically control the Lifecycle State of a given Item (also referred to as the Controlled Item). Lifecycle Promotions are defined for an Activity in a Workflow and are triggered when a specified event occurs, such as **On Activate**. When a Lifecycle promotion is initiated, the system automatically promotes the Controlled Item's Lifecycle State from its current state to another state in its Lifecycle map. Lifecycle promotions are Relationships of a Workflow Activity. Use the following fields to configure a Lifecycle promotion from a Workflow:

①	**Event**	There are many server events that can be used to trigger the promotion, such as On Activate, On Close, On Assign, On Refuse, etc. On Activate will initiate the promotion when the activity becomes active.
②	**Item Type**	The ItemType of the Controlled Item.
③	**Lifecycle Map**	The Controlled Item Lifecycle map to be used in the promotion.
④	**From State**	The current state of the Item.
⑤	**To State**	The state the Item is to be promoted to.
⑥	**Role**	The identity defined on the transition in the Lifecycle map. However, a promotion that is triggered by a workflow activity bypasses the role defined on the transition. The promotion occurs based upon the person voting on the activity and the specified event.

Promotion Guidelines

- Current Lifecycle state of the Controlled Item must match one of the “From State” values
- Multiple promotion configurations are supported
- Only promotes states for current Controlled Item
- Workflow Activity assumes the Role is provided through the Promotions configuration (bypasses User Identity)

aras

5 | © 2023 Aras

Promotion Guidelines

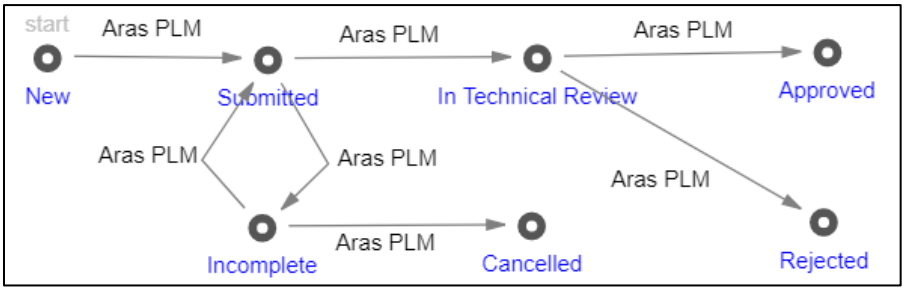
The Controlled Item’s current Lifecycle State must match one of the defined Lifecycle Promotion’s **From State** values when the promotion is initiated. If no matching From State or Lifecycle transition is found, the system rolls back to the prior Workflow Activity and displays an error on the Voting Dialog. For example, if the Controlled Item is currently in the **Draft** state and an Activity with a Lifecycle Promotion is initiated, there must be at least one Promotion defined for the Activity which has a **From State = Draft**.

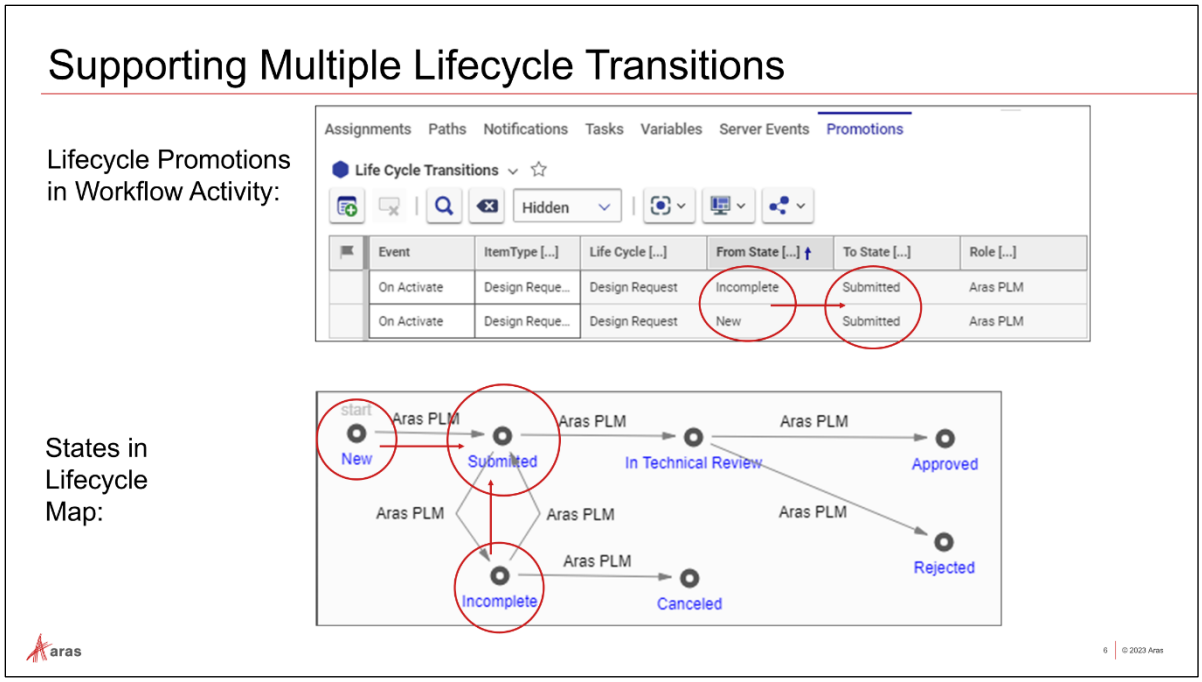
A Lifecycle Promotion from a Workflow cannot be used for promoting ItemTypes that are not defined as the **Controlled Item** for the Workflow. A good practice is to use the Aras PLM identity as the Role on the Transition in a Lifecycle map that will be initiated from Workflow Activities.

Try it … Change the Lifecycle Map

Use the Aras PLM Identity to disallow manual promotions.

1. Open the **Design Request** Lifecycle map for editing.
2. Select each transition and change the role to Aras PLM.
3. Click Done for the Lifecycle map.





Supporting Multiple Lifecycle Transitions

It is common to have multiple transitions to each state in a Lifecycle map. Therefore, the workflow activity must be configured to handle all incoming State promotions. For example, looking at the Lifecycle map above, you notice two transitions are entering the **Submitted** State, one from **New** and + the other from **Incomplete**. The Controlled Item's current Lifecycle state must match one of the defined Lifecycle Promotion's **From State** when the promotion is initiated or else it will fail and display an error on the Voting Dialog.

Two promotions must be set on the **Review Request** activity on the workflow map. A Design Request (the controlled item) needs to be promoted to the Submitted State when it reaches the Review Request Activity in the Workflow. The Review Request Activity can be reached from the Verify Request Activity (New lifecycle state) or Provide Missing Info Activity (Incomplete lifecycle state). This means a controlled item can transition to the Submitted state from two different states (**New** and **Incomplete**). Failure to provide both promotions results in an error message in the Voting Dialog window.

Try it … Configure a New Lifecycle Promotion

1. Open the **Design Request** Workflow Map for editing.
2. Select the **Review Request** Activity in the Workflow Canvas, locate the Promotions relationship tab, and add a Promotion relationship.
3. In the Select Items dialog – Life Cycle Transitions window, type in ***Design Request*** in the Life Cycle […] search criteria column and click Run Search.
4. Select the appropriate promotion From and To State from the available lifecycle transitions. In this example, select the promotion **From State: New → To State: Submitted** and click the OK button to return to the Promotions tab.
5. On the Promotions tab, click in the ItemType […] field, find and select the **Design Request** ItemType.

6. Continue adding Lifecycle Promotions using the following table:

	Promotions		
Activity	**From State**	**To State**	**Role**
Review Request	Incomplete	Submitted	Aras PLM
Perform Tech Review	Submitted	In Technical Review	Aras PLM
Provide Missing Info	Submitted	Incomplete	Aras PLM
Cancel Request	Incomplete	Cancelled	Aras PLM
Approve Request	In Technical Review	Approved	Aras PLM
Reject Request	In Technical Review	Rejected	Aras PLM

7. Click Done for the Workflow map.

Note

If you get an error when saving your Workflow Map, check to make sure that each of your promotions have the ItemType value filled in with the value: *Design Request*.

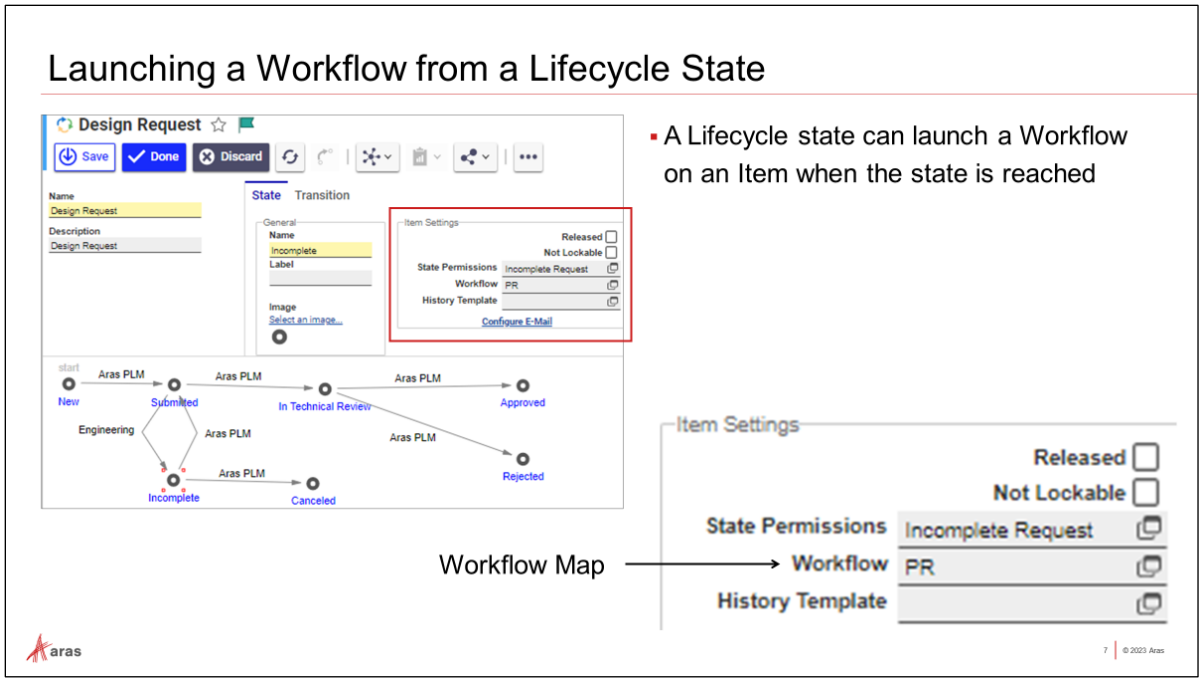
Try it … Test the Workflow

In this exercise, you test the workflow by creating a new Design Request Item and completing the Activities in the InBasket.

Note that some of the Activities will not have tasks, because we did not create them for all the Activities.

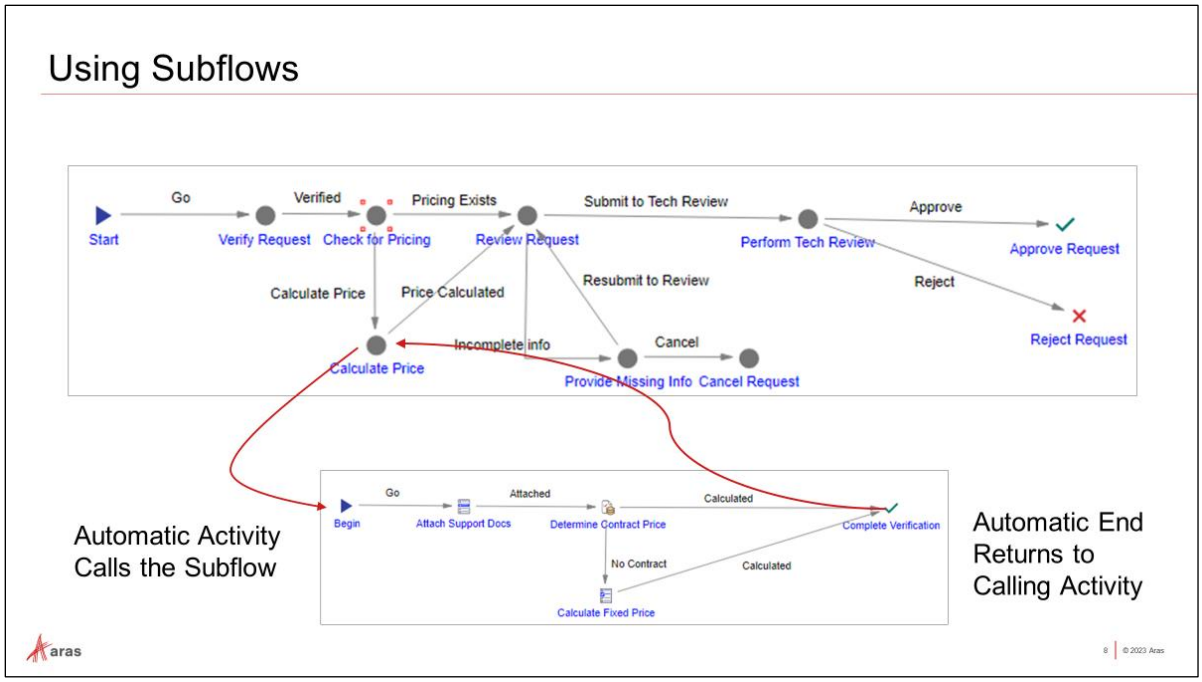
1. Sign in as Diane Prescott.
2. Create a new Design Request, assigning Diane Prescott as the Request Owner, and complete the Activities in the InBasket.
3. Notice that the Lifecycle State of the Design Request automatically promotes after completing each workflow assignment.
4. Use the following table as a reminder of the assigned Identities.

Identity	Members
CRB	Diane Prescott, Dan Park (and Anne Reddy, added via the labs)
Engineering	Aaron Lee, Diane Prescott, Jim Hance, Mike Miller, Terry Adams, John Young, Peter Smith, Susan Harris



Launching a Workflow from a Lifecycle State

You can set a Workflow to initiate when a Lifecycle State is activated. This is a good option if you do not want the Workflow process to start at Item creation.



Using Subflows

Any workflow may be added as a subflow to any other Workflow to enable a process reuse or to clarify the graphical display.

The parent Workflow continues when the subflow is complete.

Subflows allow for a Workflow hierarchy by creating a form of branching. The current Workflow activity is suspended until the Subflow completes and returns control. A Subflow can be used with branching, because only the Activity containing the Subflow suspends, not the whole Workflow Process.

In this example, the Calculate Price Activity calls a separate Workflow, as a Subflow, to step through the process of collecting Support Documents and determining a Price. When all the steps in the Subflow are complete, the main workflow activity continues.

Notes

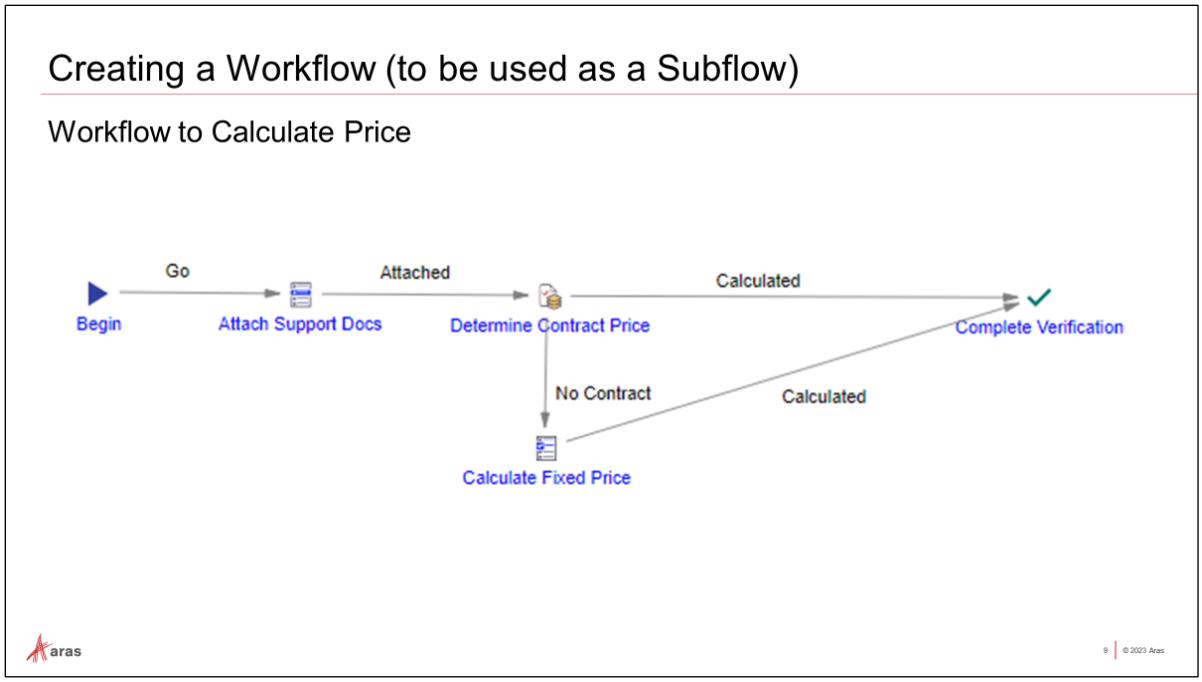
Activities with Subflows should always be set to Automatic.

In the main workflow map, the path following the automatic activity should be a Default path.

The End Activity on the Subflow should always be set to Automatic to return to the calling Activity.

Subflows must also be assigned workflows for the ItemType involved.

The Subflow must not be set as the Default workflow on the ItemType.



Creating a Workflow (to be used as a Subflow)

A subflow is simply a workflow that is called, or started, from an activity in another workflow. The calling activity waits for the called workflow (the subflow) to complete before continuing.

Try it … Create the Workflow

Use the following exercise to create a workflow that calculates a fixed price.

1. Navigate to Administration > Workflow Maps and create a new Workflow Map.
2. Provide the Name, Description, and Process Owner.

   In this example, use the following values:

Name	Calculate Price
Description	Collect information to establish pricing
Process Owner	Administrators

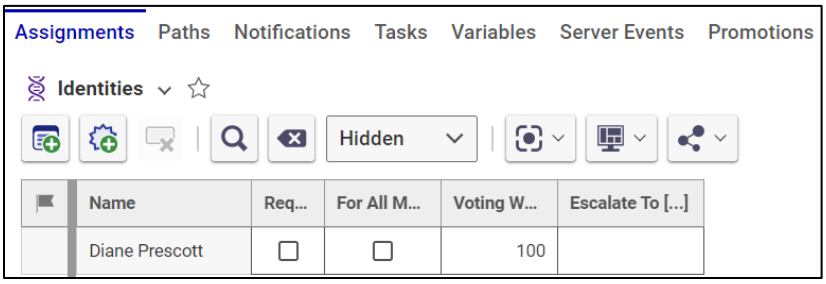
3. Save the new workflow map.
4. Rename the Start activity *Begin* and the End activity to *Complete Verification*.
5. Add the following Activities to the Calculate Price Workflow Map. You can change the activity icons, if desired:

Attach Support Docs
Determine Contract Price
Calculate Fixed Price

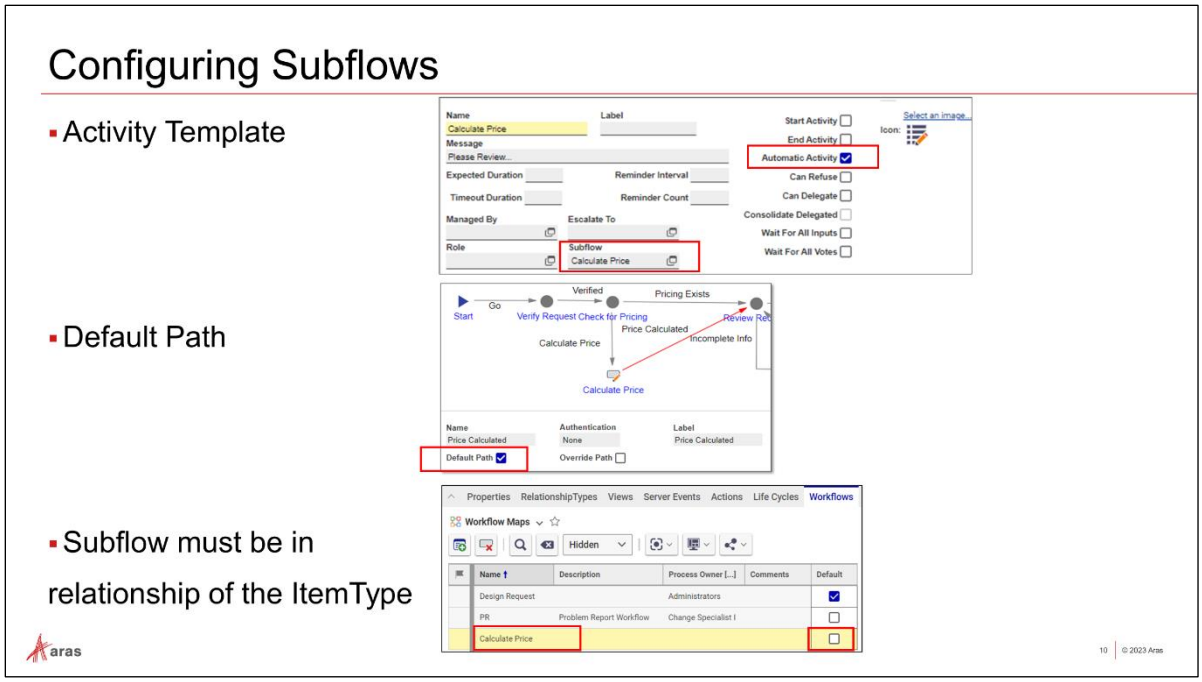
6. Add Workflow Paths with the suggested settings as shown below:

From Activity	To Activity	Path Name	Options
Begin	Attach Support Docs	**Go**	Default Path = True
Attach Support Docs	Determine Contract Price	**Attached**	
Determine Contract Price	Complete Verification	**Calculated**	
Determine Contract Price	Calculate Fixed Price	**No Contract**	
Calculate Fixed Price	Complete Verification	**Calculated**	

7. For training purposes, we will simplify it by sending each Assignment to Diane Prescott. On the three new activities (**Attach Support Docs, Determine Contract Price, and Calculate Fixed Price**); therefore, configure an Assignment to the **Diane Prescott** identity with a Voting Weight of *100*.



8. Ensure that both the Begin and Complete Verification activities remain configured as Automatic.
9. Feel free to change the icons, if desired.
10. Click Done for the Calculate Price Workflow Map and close the tab.



Configuring Subflows

A called workflow is also referred to as the subflow. To configure a subflow into a main workflow, first edit the relevant Activity Template on the main workflow and specify the subflow to be called ; ensure the *Automatic Activity* setting for that calling activity is set to true. Then, the path exiting that calling activity (here 'Price Calculated ') must be set as the Default Path. Finally, you must edit the ItemType definition to include the subflow as one of the workflow maps for the item.

Try it … Configure a Workflow to Call a Subflow

1. Navigate to Administration > Workflow Maps and open **the Design Request** Workflow Map for editing.
2. Select the path between the **Verify Request** and **Review Request** activities, right-click, and choose Insert Activity from the context menu.
3. Name the new Activity *Check for Pricing*.
4. Below the **Check for Pricing** activity, add another new activity named *Calculate Price*.
5. Edit or create Workflow Paths with the suggested settings as shown below:

From Activity	To Activity	Path Name	Options
Check for Pricing	Review Request	**Pricing Exists**	
Check for Pricing	Calculate Price	**Calculate Price**	
Calculate Price	Review Request	**Price Calculated**	Default Path = True

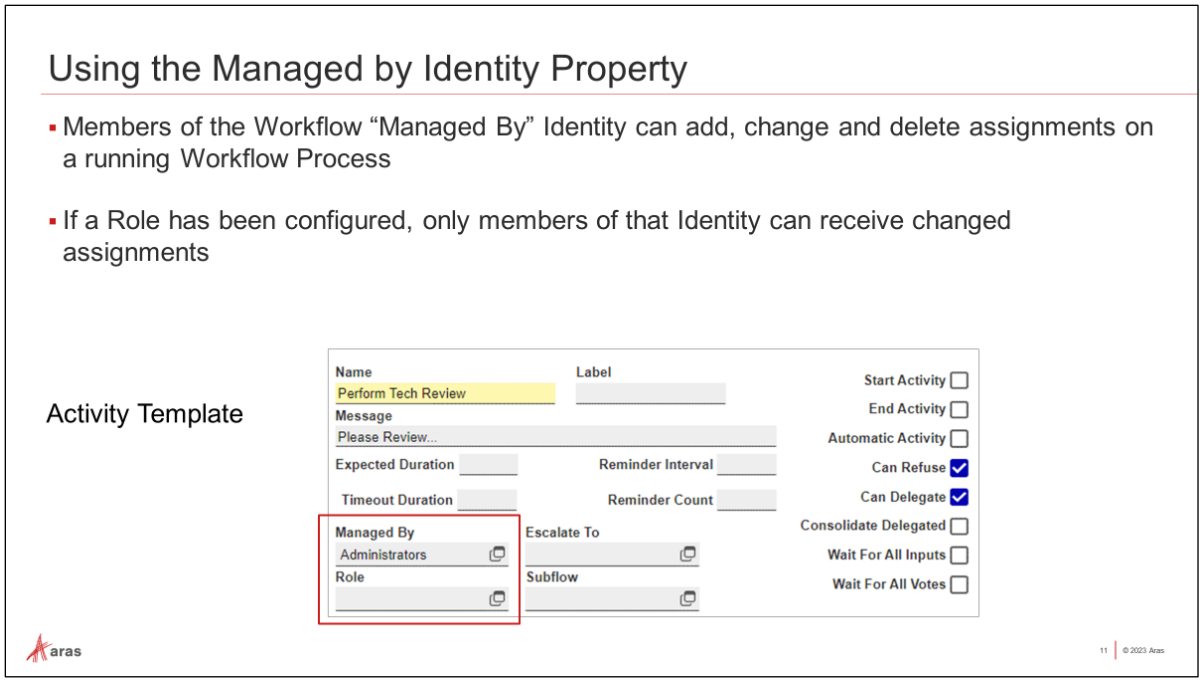
6. On the Check for Pricing activity, create an assignment to **Diane Prescott** with Voting Weight of *100*.
7. On the Calculate Price activity, call a subflow by selecting the **Calculate Price** workflow.

8. Set the **Calculate Price** activity to be an Automatic Activity, uncheck Can Refuse and Can Delegate, and ensure the **Price Calculated** path is set as a Default Path.
9. Click Done for the Design Request Workflow Map.

Try it … Configure the ItemType and test

1. Navigate to Administration > ItemTypes and open the **Design Request** ItemType for editing.
2. On the Workflows tab, add the **Calculate Price** Workflow Map as an assigned workflow, and click Done for the **Design Request** ItemType.
3. Sign in as Diane Prescott (dprescott/innovator) and test the workflow with the subflow by creating a new Design Request with Diane Prescott as the Request Owner.
4. Process all activities and be sure to vote **Calculate Price** on the Check for Pricing activity to start the subflow.
5. Continue processing the Design Request until your InBasket is clear.

   Note: the subflow returns to the main workflow where the next Activity **Review Request** is assigned to the Engineering identity.



Using the Managed by Identity Property

The **Managed By** Identity has the authority to edit the assignment of an active workflow process. Examples of changes could be a dynamic assignment, replacing an assignee, adding, or removing assignees, and manipulating the voting weights. In the example above, Administrators have the right to change the current assignment(s) on the Perform Tech Review workflow activity in a running Workflow process.

To make a Dynamic Assignment using the **Managed By** property, you leave the Assignment blank for the Workflow Activity, thus forcing an assignment on demand.

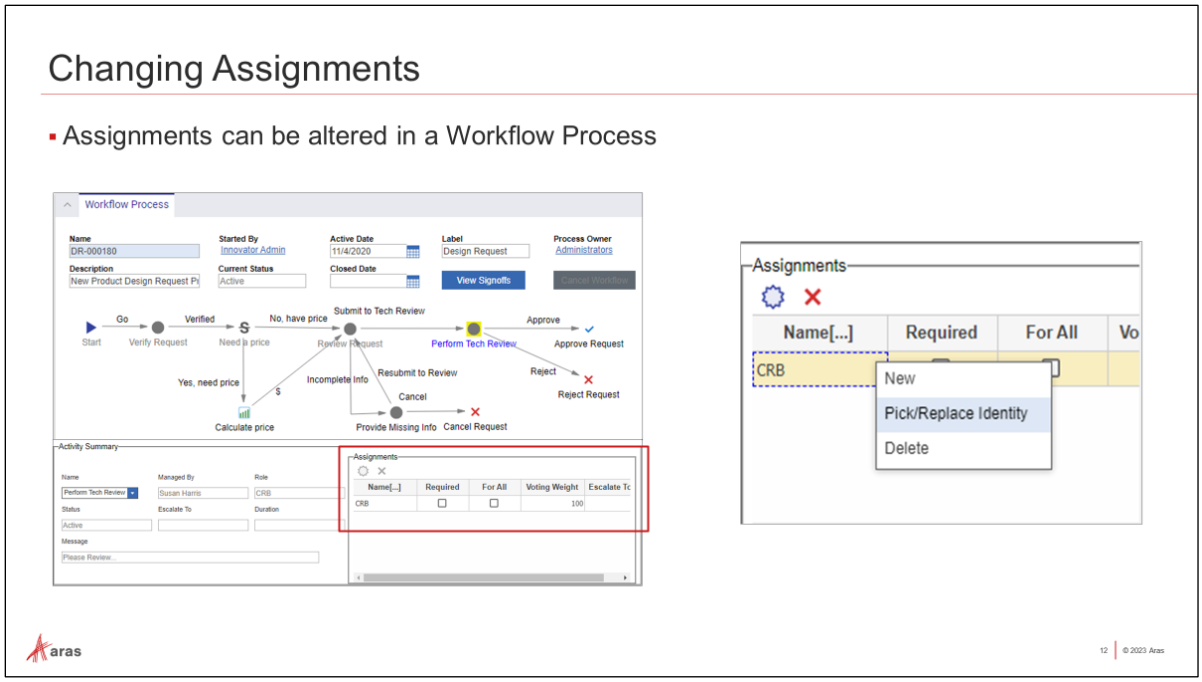
Assignment Rules

The **Managed By** property must be set in advance on the Workflow Map so that assignments can be changed on active workflow processes.

If you want to limit the assignment choices, the Role property must be set. When the Role property is set, the Activity returns an error if you try to assign an Identity that is not a member of the Role Identity property.

Try it … Configure the Managed by Property

1. As an Administrator, navigate to Administration > Workflow Maps and open the **Design Request** Workflow Map for editing.
2. On the **Perform Tech Review** activity, set the Managed By Identity to **Administrators** and leave the Role field empty.
3. Click Done for the Workflow map.



Changing Assignments

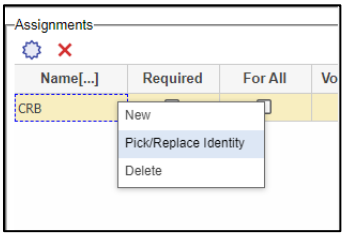
If the **Managed By** Property is set, Assignments can be altered in the Assignments section of the workflow process dialog.

Try it … Change Assignments

1. As Diane Prescott, navigate to Product Management > Design Request, and create a new Design Request item with Diane Prescott as the Request Owner.
2. Click Done and note the DR- number.
3. In your InBasket, process each workflow assignment (Vote) until the Design Request Workflow process is in the Review Request activity.

   The simplest path is **Pricing Exists**.
4. Still as Diane Prescott, a member of the **Engineering Identity**, go to the InBasket, process the Design Request to Perform Tech Review.

   (Vote Submit to Tech Review.)
5. As an Administrator, search for and open the Design Request item you processed through to the Perform Tech Review workflow activity.
6. On the Navigate menu, select Workflow to display the workflow process diagram.
7. Click the Edit button to make changes to the active assignment.
8. In the Assignments section of the window, right-click on the current assignment (**CRB**) and select **Pick/Replace Identity** from the context menu.



9. Search and select an identity to Perform the Tech Review.
10. Click Done.
11. Login in as the identity you chose to Perform the Tech Review, open the InBasket, and either Approve or Reject the Design Request.

Summary

In this unit, you learned how to enhance a Workflow.

You should now be able to:

- Promote an Item from a Workflow
- Launch a Workflow from a Lifecycle State
- Create and call a Subflow
- Change Assignments on Active Workflows

Review Questions

1. If a Workflow promotes an Item to a new state, what Identity is used to process the promotion?

2. What Activity setting is important when using a subflow from an Activity?

3. How do you limit what Identities can be assigned using the Managed By property?

Lab Exercise

Goal

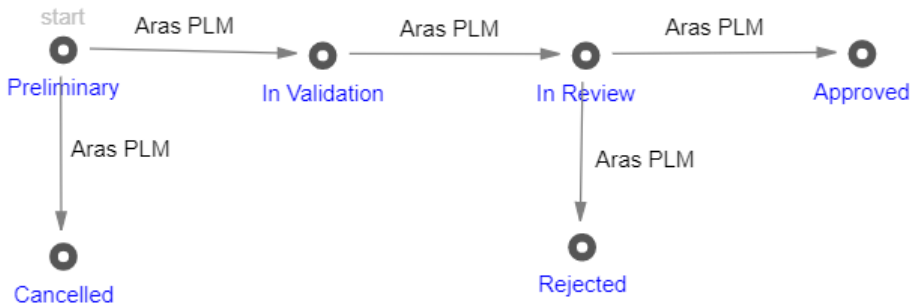
Be able to integrate an Item Lifecycle map with a Workflow.

Scenario

In this exercise, you add promotions to the current Change Request Workflow to allow the Workflow to control the state of an Item.

Steps

1. Open the Change Request Lifecycle map for editing.
2. Select each transition and change the role to Aras PLM.



3. Click Done for the Lifecycle map.
4. Open the Change Request Workflow Map for editing.
5. Set the following promotions for each Workflow activity as follows:

Activity Name	Event	ItemType	From State	To State
Cancel Request	On Activate	Change Request	Preliminary	Cancelled
Branch	On Activate	Change Request	Preliminary	In Validation
Review Request	On Activate	Change Request	In Validation	In Review
Approve Request	On Activate	Change Request	In Review	Approved
Reject Request	On Activate	Change Request	In Review	Rejected

6. Save your changes.
7. Create a new Change Request Item, and as you process each assignment, check the state of the Change Request. You should see the Item change state based on the promotions you have set above.

Add a Subflow

1. Create a new Workflow Map named Quality Control
   (Process Owner: Administrators) using the following diagram:

2. Assign the **Quality Checks** Activity to Administrators and create a required Task for the assignee to perform standard quality control checks.
3. Click Done for the Quality Control workflow.
4. Open the Change Request Workflow and assign the **Quality Control** workflow as the subflow of the **Verify Request** activity.
5. Make this **Verify Request** activity automatic and remove its assignment, as the subflow will control the Assignment.
6. Set the **Submit for Review** path as the Default Path and then click Done for the Change Request workflow.
7. Open the **Change Request** ItemType and add the **Quality Control** workflow to the ItemType (non-default).
8. Login as a member of the **Engineering** identity (Aaron Lee, Diane Prescott, Jim Hance, Mike Miller, Terry Adams) and create a new Change Request assigning a member of the **Engineering** identity as the Owner.
9. Process the Change Request item through to approved and keep in mind the following Activity identities:
   a. The **Engineering** identity you used as the Request Creator, will need to Confirm the Request and Submit it.
   b. **Change Specialist I** (Terry Adams) will need to Validate the request.
   c. Verify Request goes through the Quality Control subflow (Administrators).
   d. Finally, Review Request to Approve or Reject is voted on by the CRB group identity (Ann Reddy, Dan Park, Diane Prescott).

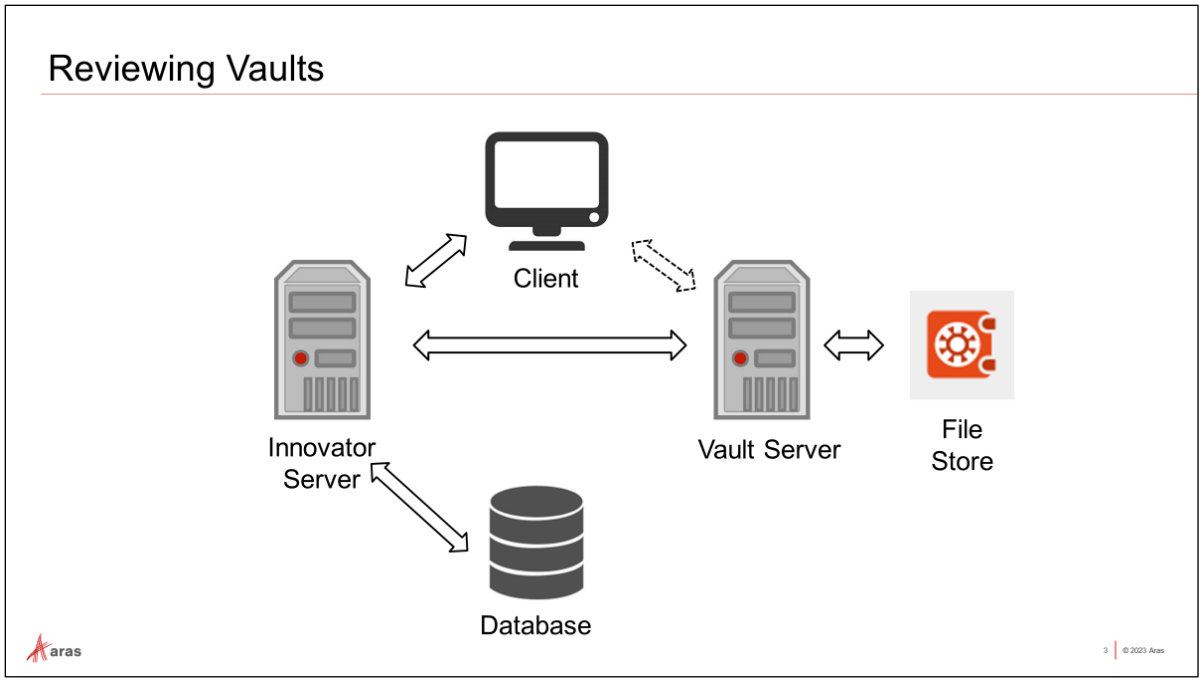
Unit 16 Working with Files

Overview

In this unit, you learn how a vault is configured to store files uploaded by users. You learn how to assign a User to a vault as well as learn about the association of File Items to a container Item. You also learn how to configure an ItemType or Relationship to support file upload/download.

Objectives

- Understand where files are stored
- Define the purpose of a File Item
- Upload and Download files
- Understand the Options for Configuring File Items
- Configure an ItemType to Support File Items



Reviewing Vaults

Files are stored in a vault (a secure file store - not the SQL database) and an IIS Vault Server application is dedicated to each vault that is configured. The Vault Server communicates with the Aras Innovator Server to keep track of files and their locations (file meta-data) in the database.

When Aras Innovator is installed, a vault named **Default** is created to store user files. You can configure additional vaults if necessary. Steps to install more than one vault are described in the *Aras Innovator Installation Guide*.

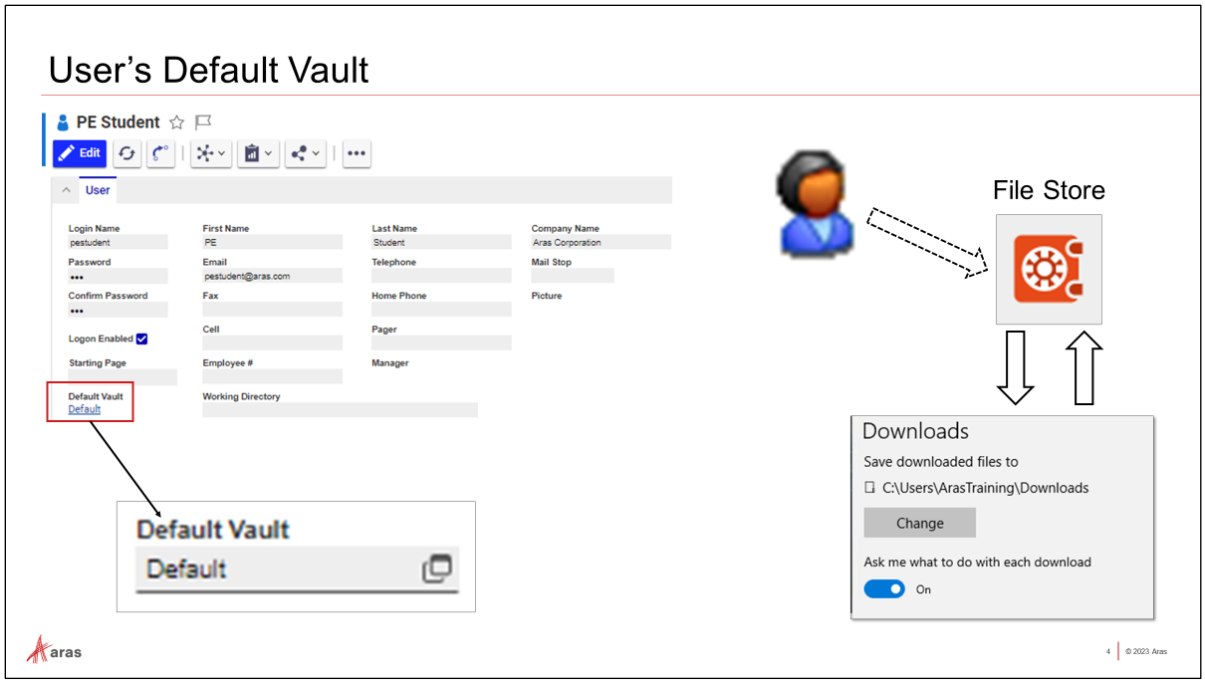
Reviewing the Aras Innovator Vault

When Aras Innovator is installed, a **Default** vault is automatically created and configured for the system. A vault contains a URL address to the Vault Server.aspx file used for processing files into the vault.

The Vault Server

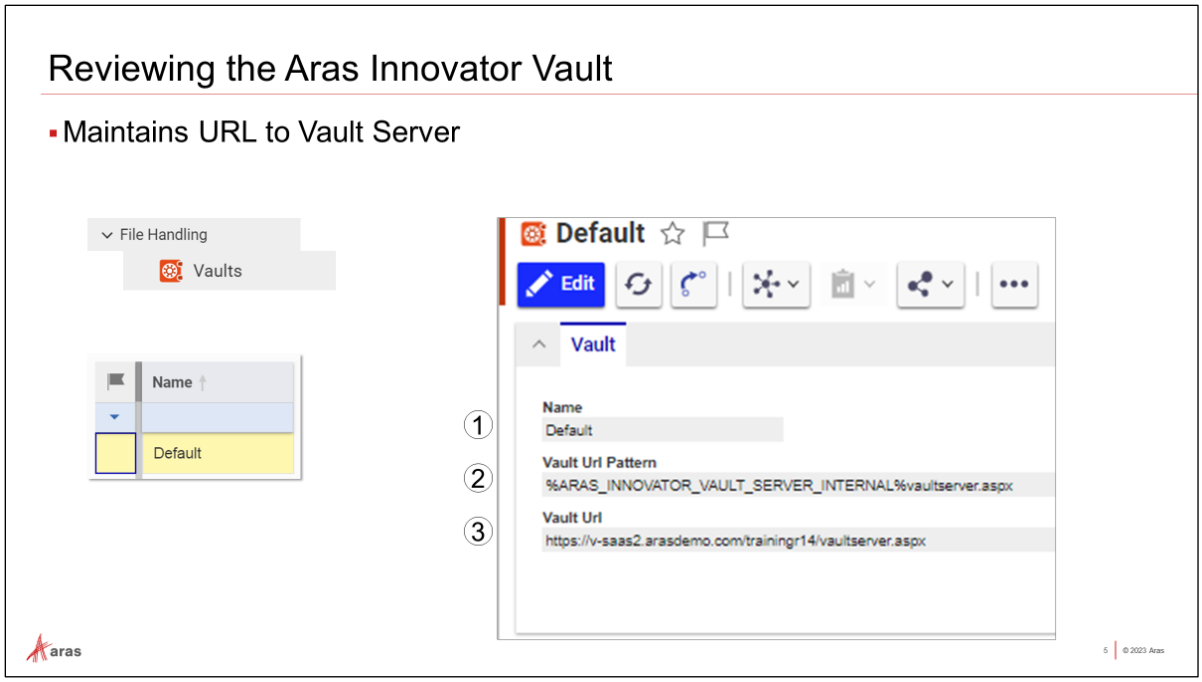
Configure the Vault Server application using the VaultServerConfig.xml file, which contains the appropriate URL for the Aras Innovator server as well as the matching vault name. The following keys are defined in the file:

- AppServerURL – The URL of the installed Aras Innovator application.
- LocalPath – The top-level directory used as the file store.
- name – The name of the vault.
- login_name – The system user that has access to the Vault Server application. (vadmin is the default user).
- password – The password of the login user defined above.
- version – Aras Innovator version number.



User's Default Vault

The Default Vault is set and managed on the User Item. The preset assignment is to the vault named **Default**. When the user uploads a file, the default vault server processes the request and saves the file. With the right permissions, a user can get a file from any vault.



Reviewing the Aras Innovator Vault

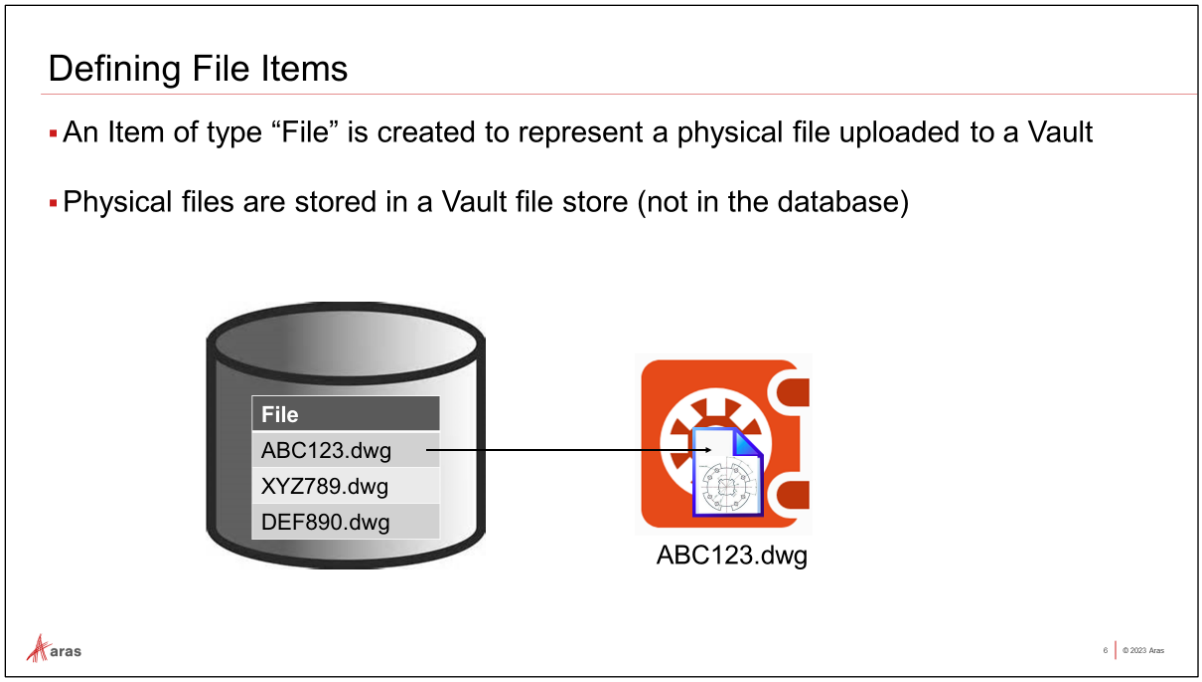
The vault Item maintains a URL to the Vault Server.aspx location, which contains the logic to copy a file from a user's client machine to a file server directory store. The vault Item is then assigned to each User Item to determine their default vault.

The following fields are available on the vault Item, under Administration > File Handling > Vaults:

①	**Name**	The name of the vault associated with the desired Vault Server.
②	**Vault URL Pattern**	The IIS path to the VaultServer.aspx. The URL can also contain the following symbols which are substituted when the vault Item is saved: HTTP_PREFIX_SERVER = http:// or https:// HTTP_HOST_SERVER= server name. HTTP_PORT_SERVER = server port number (if exists). HTTP_PATH_SERVER = Innovator web alias specified at installation.
③	**Vault URL**	This read only field displays any substitution that occurs on the URL Pattern when the vault Item is saved.

Note

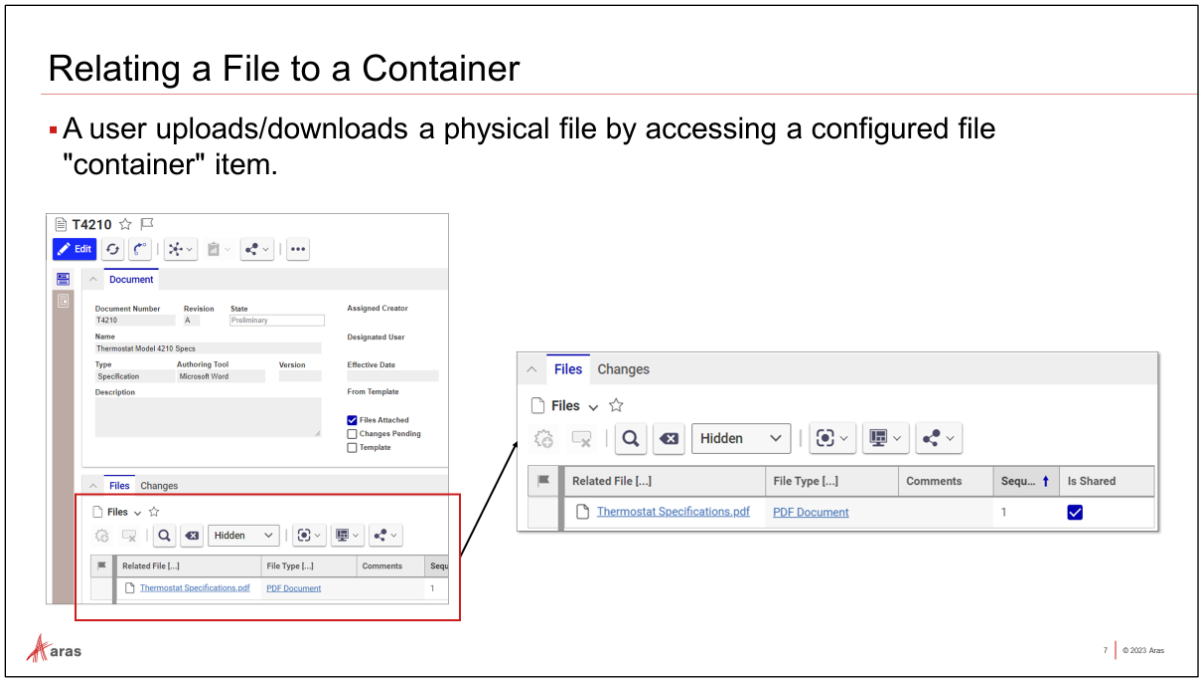
If you require more than one vault (or a custom configuration), see the *Aras Innovator Installation Guide* for more information.



Defining File Items

A physical file (e.g. Drawing, Spreadsheet, CAD File) is represented as a File Item in the Aras Innovator database. Each file that is uploaded by a user always has a reference to a File Item in the database. The File Item contains information about the file (e.g. name, created date, file size, file type).

The physical file is stored in a file store configured for each vault.



Relating a File to a Container

A user is allowed to upload (or download) a file based on a configured container item.

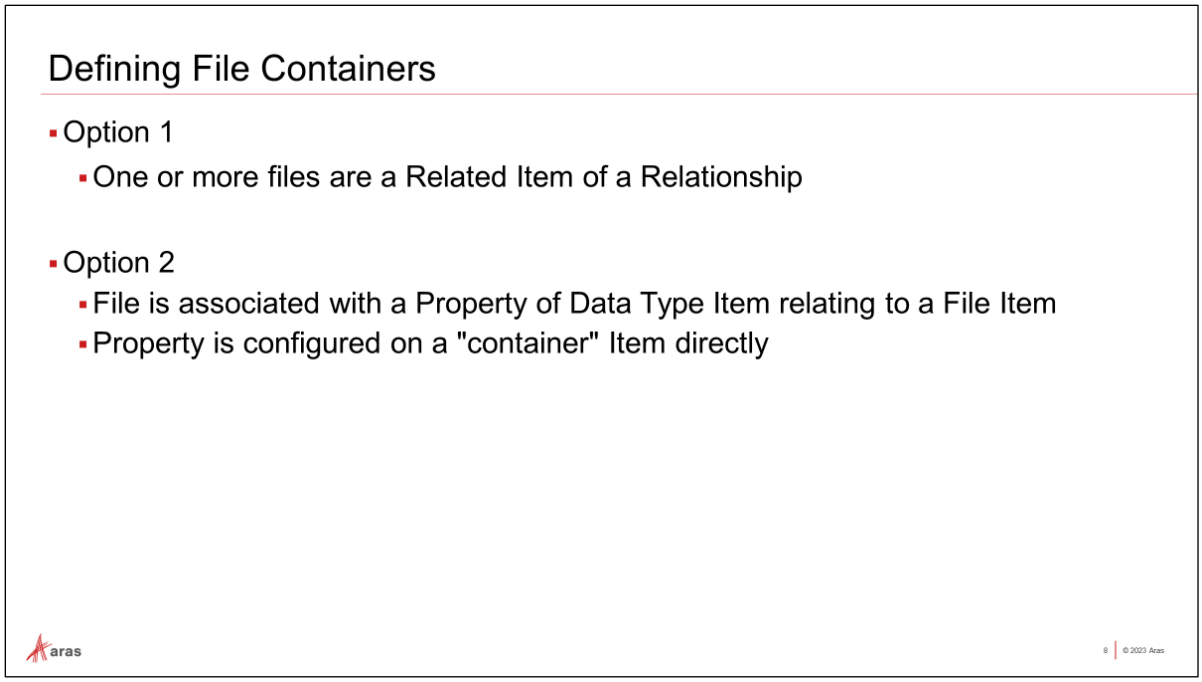
Aras Innovator provides pre-configured file containers (e.g. Document, CAD Document) or an administrator can configure additional file containers to suit yor business requirements.

Users manage files by accessing a container (based on the security of the container Item) and performing an upload from their local computers to the Aras vault or download from the Aras vault to their local computers.

In the example above, a Document Item has been created with a configured Relationship to a File Item. A user has uploaded a file named Thermostat Specifications.pdf as the related file.

Administrators can control who has access to files that have already been placed in the vault.

By default, the Is Shared checkbox is selected. This allows all users with access to the container Item to see which files are related to the container. If you uncheck it, only members of the Assigned Creator Identity for that document can see the files. This Identity is stored in the owned_by_id System Property.



Defining File Containers

There are two pre-configured options available for a user to upload/download files in the Aras Innovator database.

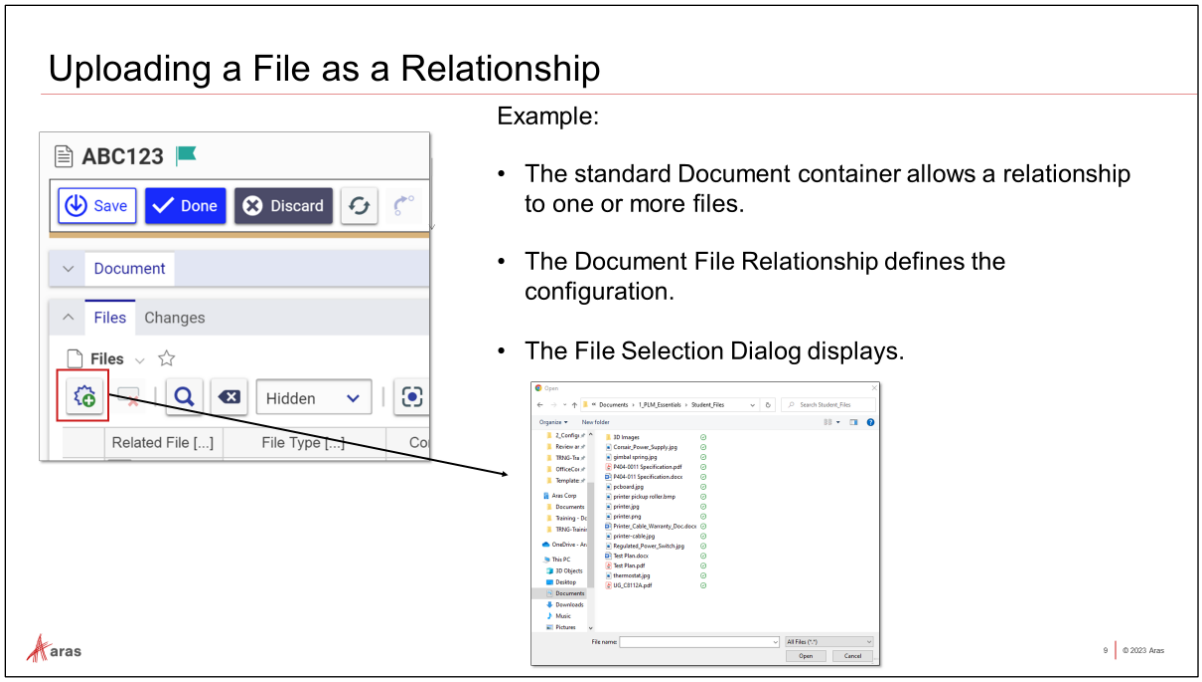
Option 1 (Document Example)

A user creates a new Relationship to one or more files on the Files tab and is prompted to select a physical file from their local computer to upload to a vault.

Option 2 (CAD Document Example)

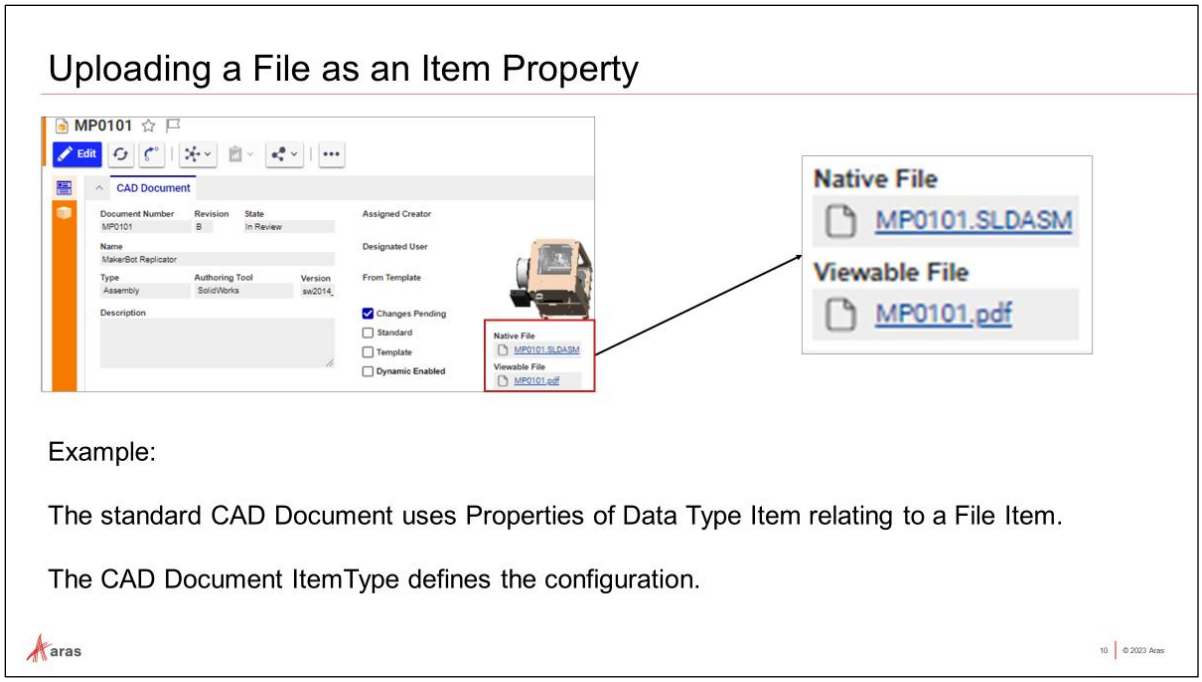
A user clicks on a Property of Data Type Item on the container Item Form to upload a physical file.

Examples of each of these two options are shown on the following pages.



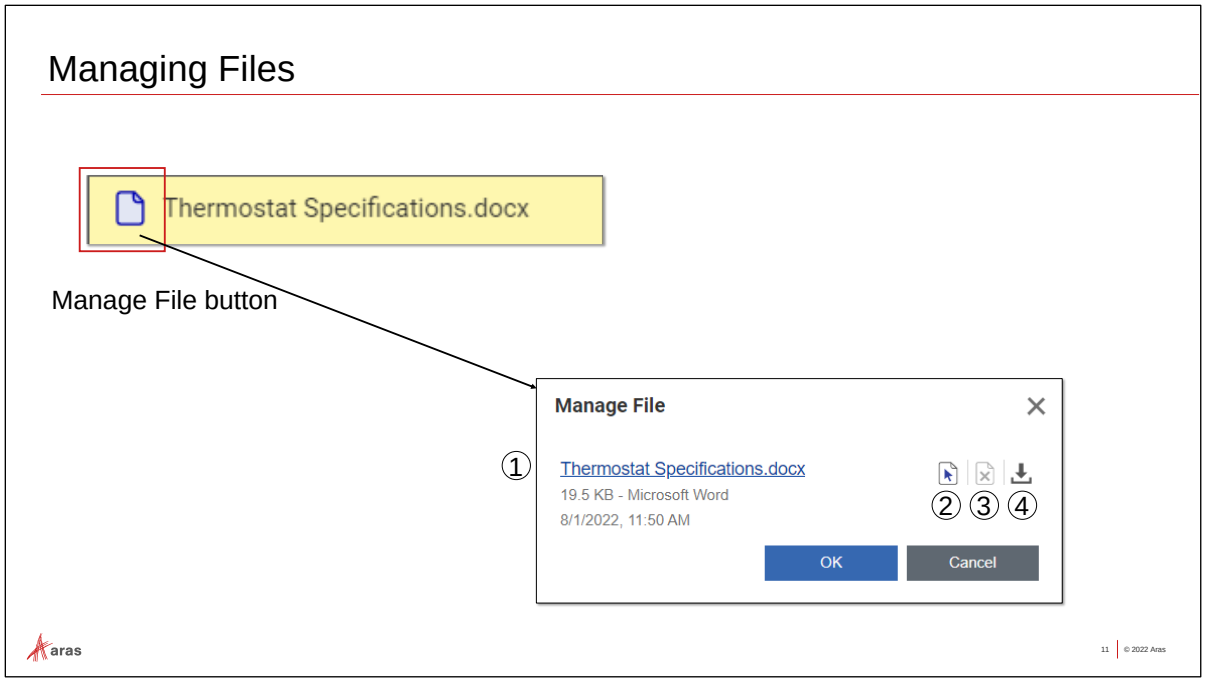
Uploading a File as a Relationship

Files added to an Item as a relationship allow for a one to many configuration. Files can be added using either the Drag and Drop method or by clicking the Add New File button. Using Add New File, shown above, requires each file to be added individually. Drag and Drop enables multiple files to be added at once.



Uploading a file as an Item Property

A file added to an item as a property has a one to one relationship. Users can either Drag and Drop the file or click the property relating to a file on a Form to upload it. The standard CAD Document Item uses this approach as shown above.

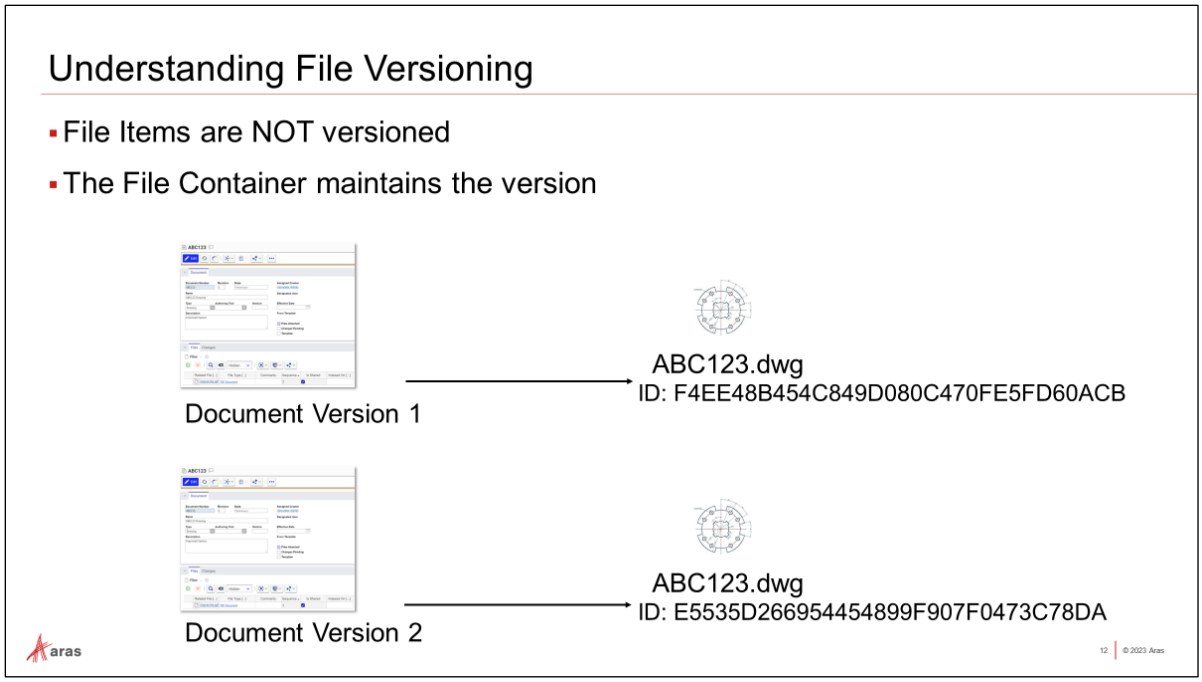


Managing Files

Once a file is uploaded to a vault, a user with the proper privileges can then manage the file using the Manage File property dialog box. A Manage File button displays to the left of the name of the uploaded file and can be clicked to display the Manage File property dialog box.

The Manage file property dialog box contains the following options:

①	**View File**	When you click the filename's hyperlink, the file is displayed (if possible) in the browser. If the file cannot be displayed, the browser prompts for a file download.
②	**Select and Upload**	Allows you to choose a physical file from your local computer and upload to a vault, replacing the previously linked file.
③	**Delete File**	Removes this file association from the current container. Note that the physical file is not deleted from the vault. The File Item is reassigned to a system container named Orphaned Files Container and is only available to the Aras Administrator.
④	**Download file**	Downloads (and/or overwrites) a file using the browser download options.



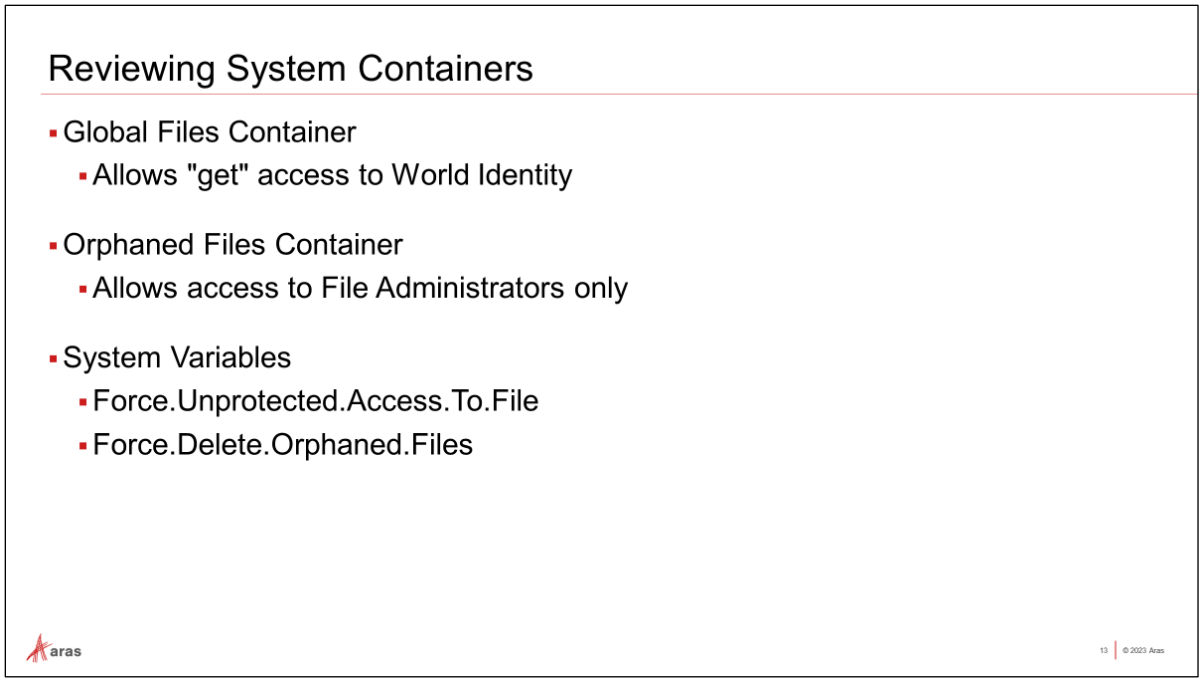
Understanding File Versioning

To simplify the management of files, a File Item cannot be versioned. The container representing the File Item represents the version of that file stored in a vault. Each time a file is uploaded to Aras Innovator a new File Item is created that represents that file.

Document Example

In the example above, a user created a new Document Item and uploaded a drawing file named ABC123.dwg. A File Item is created in the Aras database to represent this stored file.

Later, the user edits the Document Item, downloads the related file, and makes some changes to the physical file. When the user uploads the changed file to the Document Item, a new File Item is created representing that file and the version of the Document Item (container) is incremented.



Reviewing System Containers

A File Item must belong to a container item to exist in an Aras vault. A file container is an Item that contains a Property of Data Type Item with the Data Source of File. This includes Relationship Items that point to a file using the related_id Property.

Two system containers are provided if a File Item does not belong to a configured Item container. System containers are used to provide security access to a File Item that is not associated with a configured Item.

Global Files Container

If a file is created outside of a configured container by an administrator or via a program Method, the File Item is automatically assigned to the SystemFileContainer Item named Global Files Container. This allows the file **Get** access for the World Identity. As soon as a File item is related to a Property of Data Type Item in a configured Item container (e.g., a Document Item) the Permissions of the Item container replace the Global Permission.

Orphaned Files Container

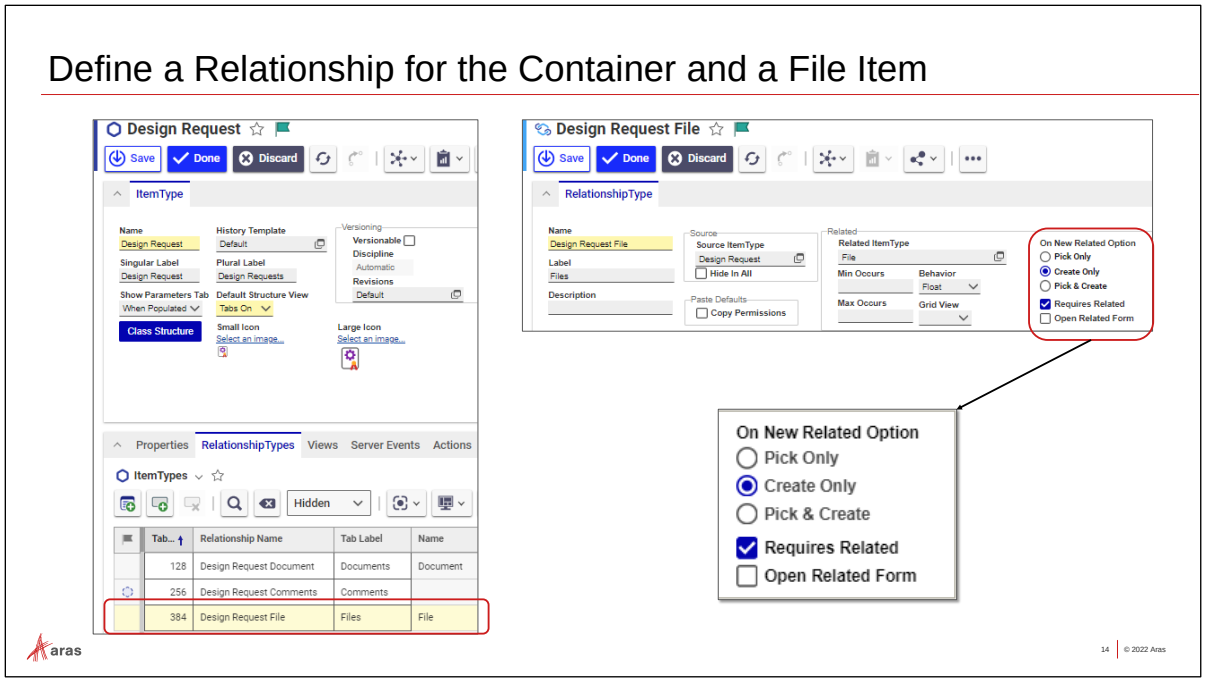
If a File Item is no longer associated with a Property of Data Type Item or a Relationship (a user replaces the File Item or deletes the configured Item container), the File Item is automatically associated to a SystemFileContainer named Orphaned Files Container. An orphaned file is available only to members of the File Administration Identity.

System Variables

Two System Variables can be created and configured to override the default File behavior:

Force.Unprotected.Access.To.File – if this variable is set to 1, all users have access to the File item and the SystemFileContainer Permissions described above are ignored. The default setting is 0.

Force.Delete.Orphaned.Files – If this variable is set to 1, any File Items assigned to the Orphaned File Container are deleted from the Aras vault. The default value is 0.



Define a Relationship for the Container and a File Item

This approach uses a Relationship that is configured with File as the related ItemType. This approach is currently used on the standard Aras Document ItemType.

Try it … Define a Relationship for the Container and a File Item

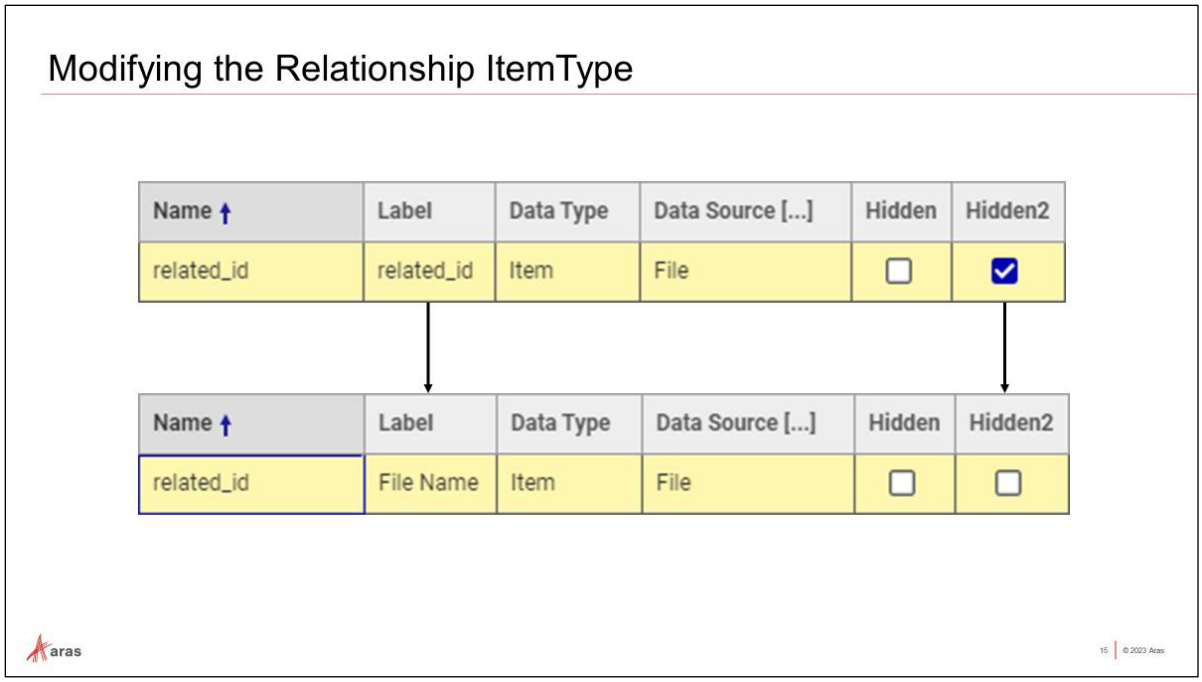
In this example, you create a new Relationship from the Design Request ItemType to the File ItemType.

1. Navigate to Administration > ItemTypes and open the **Design Request** ItemType for editing.
2. Open the RelationshipTypes tab, click the Add ItemTypes button, and in the Search dialog window, choose the **File** ItemType.
3. Type *Design Request File* for the Relationship Name and *Files* for the Tab Label.
4. Save the **Design Request** ItemType.

Try it … Modify the RelationshipType

As a standard for relationships to Files, the *Create Only* and *Requires Related* properties must be setup on the resulting RelationshipType.

1. Navigate to Administration > RelationshipTypes and open the **Design Request File** RelationshipType for editing.
2. In the On New Related Option section, click to select **Create Only**. This option prevents users from accidentally selecting a file that has already been uploaded to a vault by suppressing the Add Items button.
3. Also, click to select the **Requires Related** checkbox. This option avoids File Relationships without a connected File Item.
4. Click Done.



Modifying the Relationship ItemType

When the File Relationship ItemType is created the related_id Property is hidden by default in the Relationships grid. For the user to access the File Item from the Relationships grid, the Hidden2 option must be unchecked on the Property. If the Label is blank for a Property, the Name will be displayed as the Column Name in the grid.

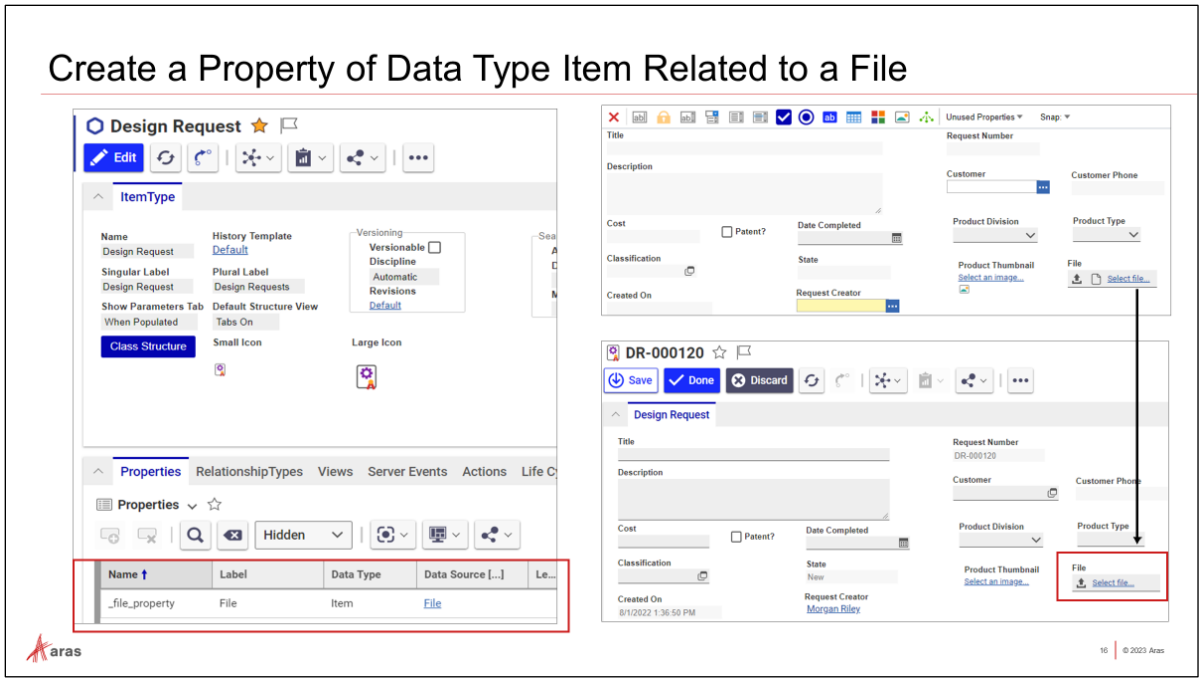
Try it … Modify the Relationship ItemType

Give the related_id property an appropriate label and make it visible in the Relationship Grid.

1. Navigate to Administration > ItemTypes and open the **Design Request File** ItemType for editing.
2. On the Properties relationship tab, locate the **related_id** property and click to clear the Hidden2 checkbox.
3. In the Label field, enter: *File Name*.
4. Click Done for the **Design Request File** ItemType.

Try it … Test the Relationship ItemType

1. Navigate to Product Management > Development > Design Requests and create a new Design Request.
2. Enter data into the required Property fields and any other desired fields.
3. On the Files relationship tab, click the New File button and upload a new file.
4. Click Done for the new Design Request.



Create a Property of Data Type Item Related to a File

This approach adds a property relating an Item of type File to the container ItemType directly (no RelationshipType). This is useful if only one file can be associated with a File container Item. This option is currently used in the CAD Document ItemType. Aras Innovator provides a custom Form control for File Items that allows for file upload and downloads.

Try it … Create a New Property

1. Navigate to Administration > ItemTypes and open the **Design Request** ItemType for editing.
2. Create a new Property, as follows:

Name	_file_property
Label	File
Data Type	Item
Data Source	File

3. Click Done.

Try it … Add the Property to a Form

1. Click the Views relationship tab and open the **Design Request** form for editing.
2. Add the Unused Property **_file_property** onto the form.
3. Click Done.
4. If need be, repeat these steps to add the same property to the other Forms related to the **Design Request** ItemType.

Try it … Test the Item Property File Relationship

1. Navigate to Product Management > Development > Design Requests and create a new Design Request.
2. On the Design Request properties form, use the **Select File** hyperlink to browse and select a related file.
3. Save and close the new Design Request.

Summary

In this unit, you learned how to work with files. You should now be able to:

- Understand where files are stored
- Associate a User to a vault
- Define the purpose of a File Item
- Upload or Download files
- Understand the Options for Configuring File Items
- Configure an ItemType to Support File Items

Review Questions

1. Files are stored in the SQL Server database. (True/False?)

2. How does a User become associated with a vault?

3. What is the purpose of the Items of type File?

4. Which Configuration Option does **not** require the use of a Relationship Item?

Lab Exercise

Goal

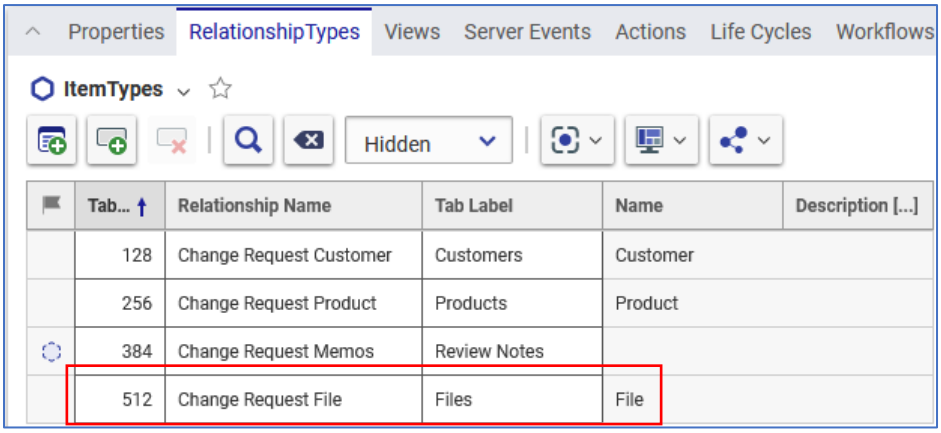
Be able to configure an ItemType to support file upload/download from the **Default** vault.

Scenario

In this exercise, you add the capability for users to upload a file to a vault directly from a Change Request. Once the file is added, a user can manage the file using the Manage File dialog box.

Steps

1. Edit the **Change Request** ItemType and add a new RelationshipType (Add ItemTypes button) named *Change Request File* that uses File as the related ItemType. Click Done.



2. Now open the *Change Request File* RelationshipType for editing and select the **Create Only** and **Requires Related** options from the On New Related Options panel. Click Done.

3. Open the **Change Request File** ItemType for editing.
4. Search for the **related_id** property and click to unselect the Hidden2 checkbox and enter File for the Label.
5. Click Done.
6. To test the new configuration, edit a Change Request or create a new one, open the Files tab, and upload a new file.
7. Navigate to Administration > File Handling > Files to view the “File” Item representing the physical file you uploaded to the Vault.

   Remember, an end user will never be allowed access to the vault directory.

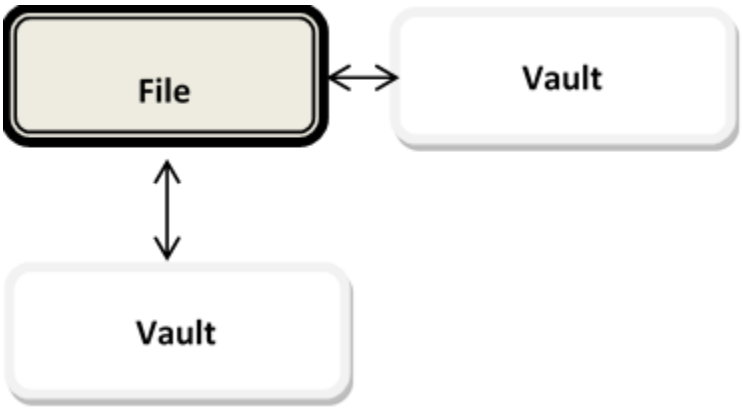
Unit 17 Replicating Files

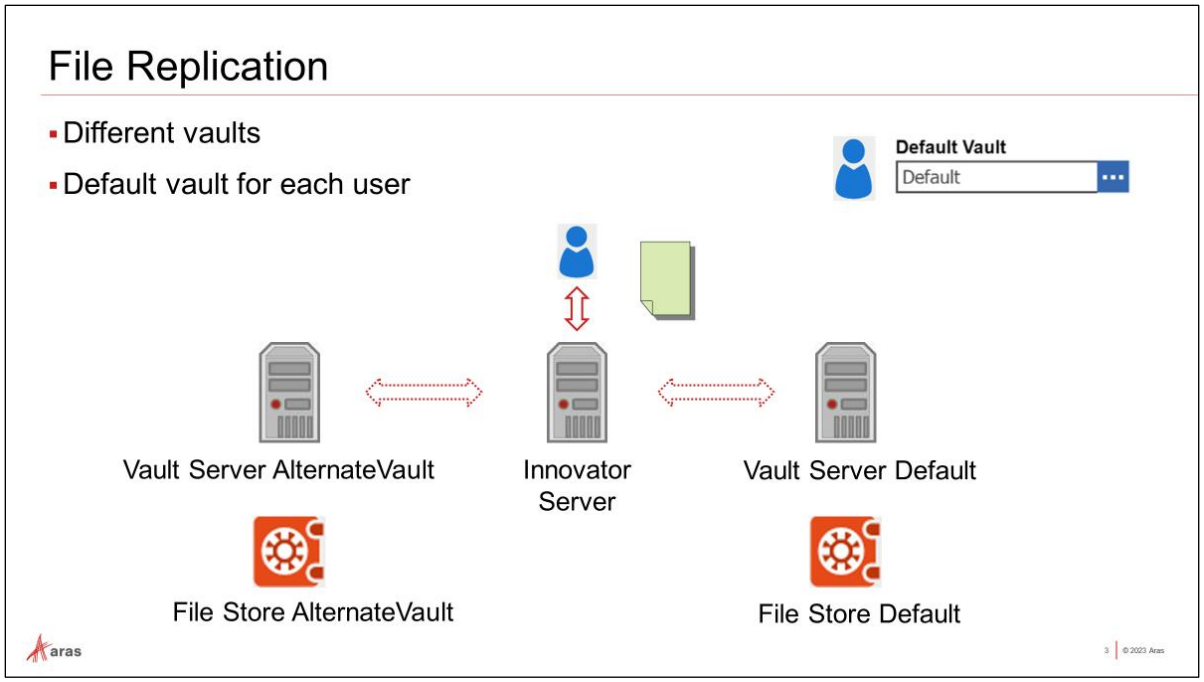
Overview

In this unit, you will learn how to configure file replication between multiple vaults. File replication is useful in enterprise installations where multiple vaults can be used to store different files based on type or size – or to allow faster retrieval of files when users are in different geographic areas. You can configure file replication in several different ways depending on the requirements of the solution. Note that the "Push" replication is only available to subscribers and requires a subscriber license key to operate. On demand replication with multiple vaults is available without subscription.

Objectives

- Create Replication Rules
- View Replication Transactions and Logs
- Describe Additional vaults
- Configure a Preferred vault
- Deactivate and Remove Replication Rules





File Replication

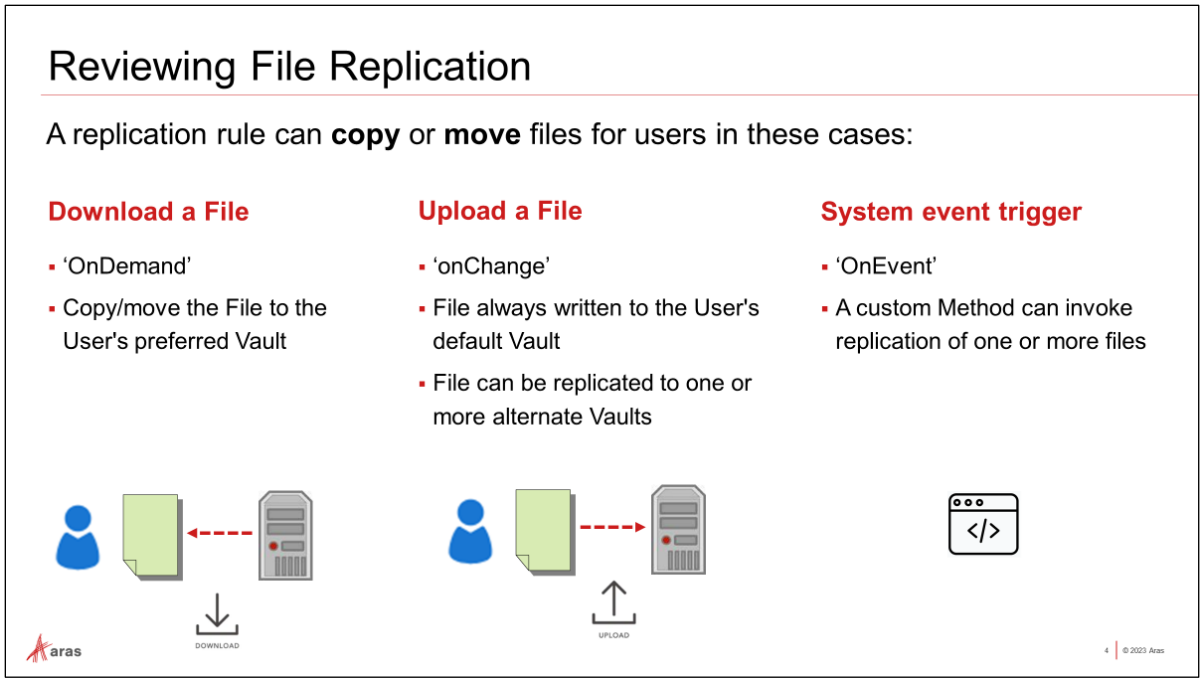
Aras Innovator uses a distributed vault environment that allows for greater flexibility.

Each user in Aras Innovator is assigned to a Default vault, where all files that they create or save are uploaded. Users are unaware of the actual vault location of files. Their access to files is controlled by the permission on the item which contains the files. When users save their file changes, their files are always uploaded to their assigned Default vault.

File replication is an asynchronous process that is invisible to the end user. Users simply locate the desired file and retrieve and edit it (based on their permissions). The system keeps track of the location(s) of the file.

When File Replication Rules are enabled, the Aras Innovator server sends execution requests to the target vault servers. The requests are acknowledged, processed by the vault server, and when complete, a message is sent back to the Aras Innovator server to be logged.

To use file replication, you need an Aras.EssentialsSubscription feature license and at least two vault servers installed and running.



Reviewing File Replication

The Replication Rules can be configured to indicate whether a file should be moved or copied to a new vault location.

The Replication Rules can be triggered in the following cases.

File download

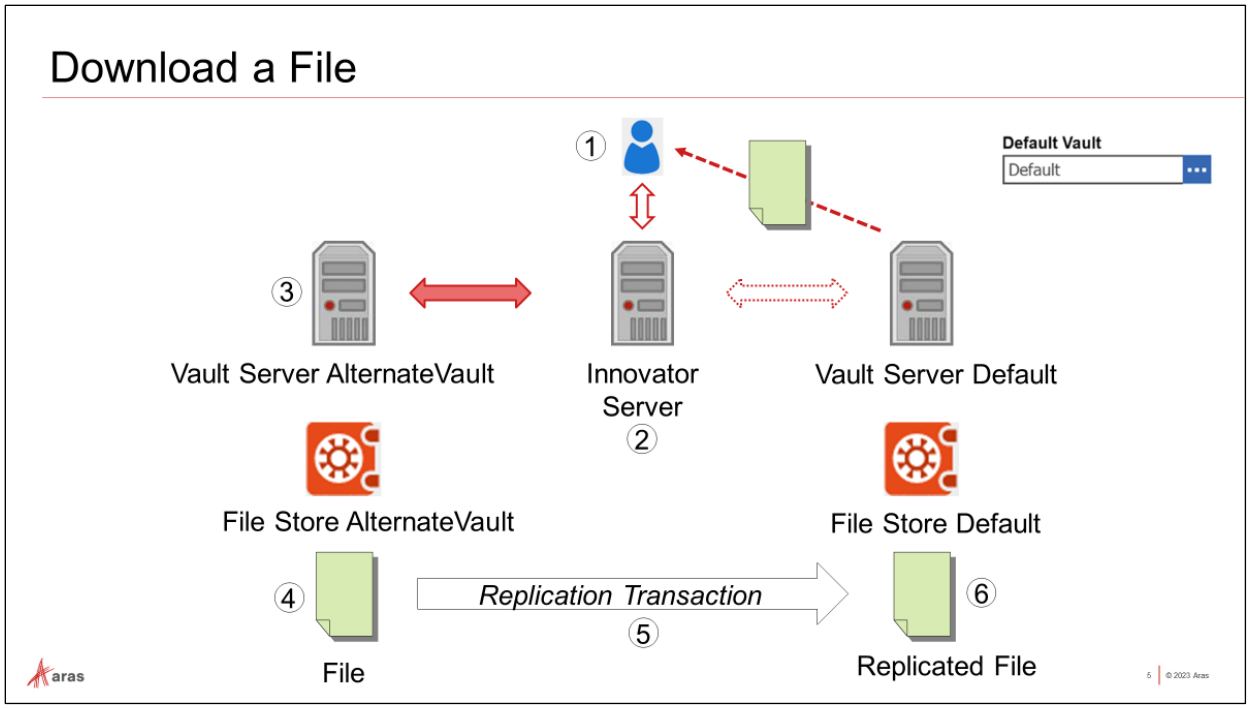
When users retrieve a file, the server sends a message to the vault storing the file to deliver a copy to the user. The vault services can be configured to dynamically locate the file and copy (or move) it to a local vault for future access.

File upload

When a user saves changes to a file and checks it into the system, the file is always written to the user's Default vault before any replication rules are checked. In addition, every vault can be configured to dynamically copy (or move) files to one or more alternate vaults.

System Event

You can create and associate a custom Method with a system event to invoke replication on one or more files in response to the event. For example, when a Document Item is promoted to the Released state, the file(s) that are related to the Document Item can be scheduled for replication to another vault.



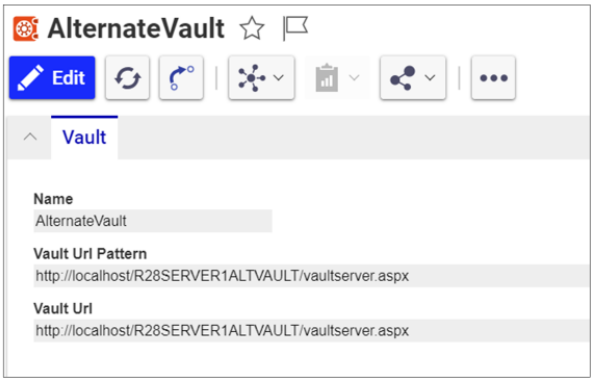
Download a File

This example shows a process for on demand file retrieval and file replication. There are many other options available. This example uses two vaults that have been configured to support file replication from the **AlternateVault** vault to the **Default** vault.

1. A User is always assigned to a Default vault in the User profile. In this example, the User's default vault is named **Default**.
2. The user selects a file to retrieve from the Aras Innovator Server. The Aras Innovator Server maintains meta-data about each file stored in a vault (actual file data is contained in the File Store). The file exists in the AlternateVault vault (and not the User's Default vault). The Aras Innovator Server sends a request to the AlternateVault vault to copy the file to the Default vault and deliver the file to the user.
3. The AlternateVault vault has been configured to replicate files on demand.
4. The file resides in the AlternateVault vault.
5. A Replication Transaction is created based on the onDemand Replication Rule on the AlternateVault vault.
6. The file is copied to the User's Default vault, based on the schedule defined in the Replication Rule, and delivered to the user.

Create Additional Vaults

- Each Vault requires a Vault Server



aras

6 | © 2023 Aras

Create Additional Vaults

To use file replication, the Aras Agent Service must be installed. IIS web services are utilized to send messages to/from the Aras Innovator server to one or more vault servers.

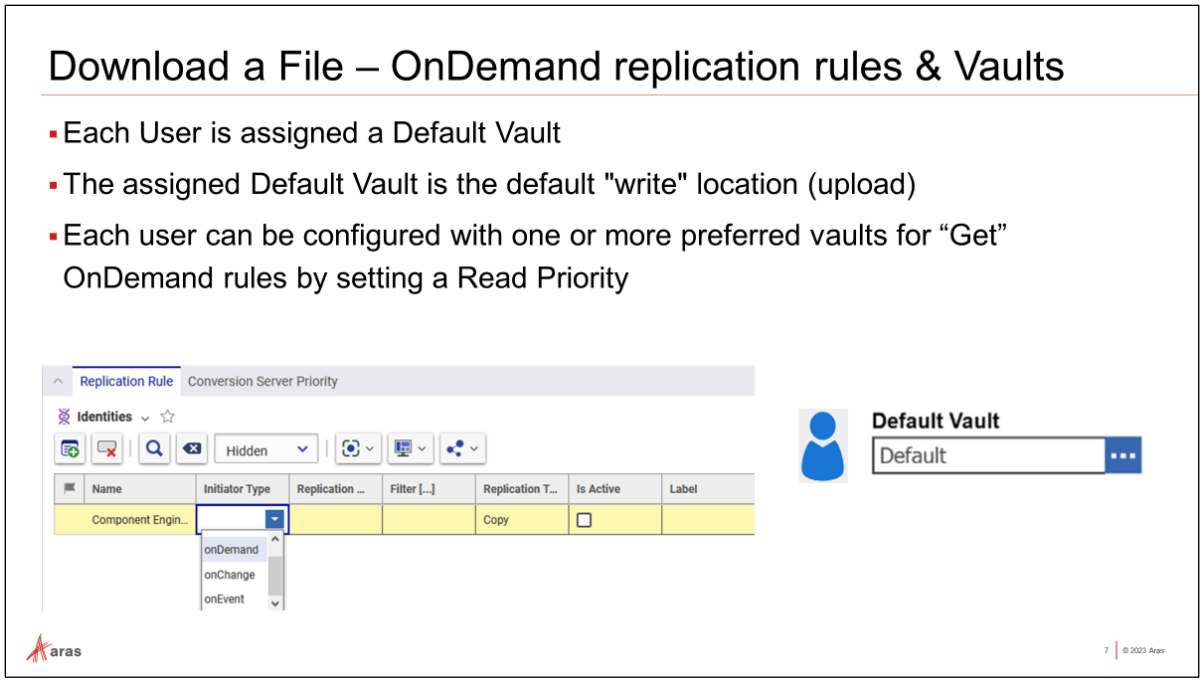
Once the vault servers are installed, additional vaults must be configured on the Aras Innovator server. This is done by creating new Vault items. There are two required properties to create a new vault.

- **Name** – New (The name of the vault server we are configuring).
- **Vault URL Pattern** – This is the IIS path to the **VaultServer.aspx**. Because this is not the default vault, the URL should be hard coded using the following pattern:
  http://<servername>/<web_alias>/vault/VaultServer.aspx

Try it … View the available Vaults

An additional vault (AlternateVault) was installed and exists in our classroom training image.

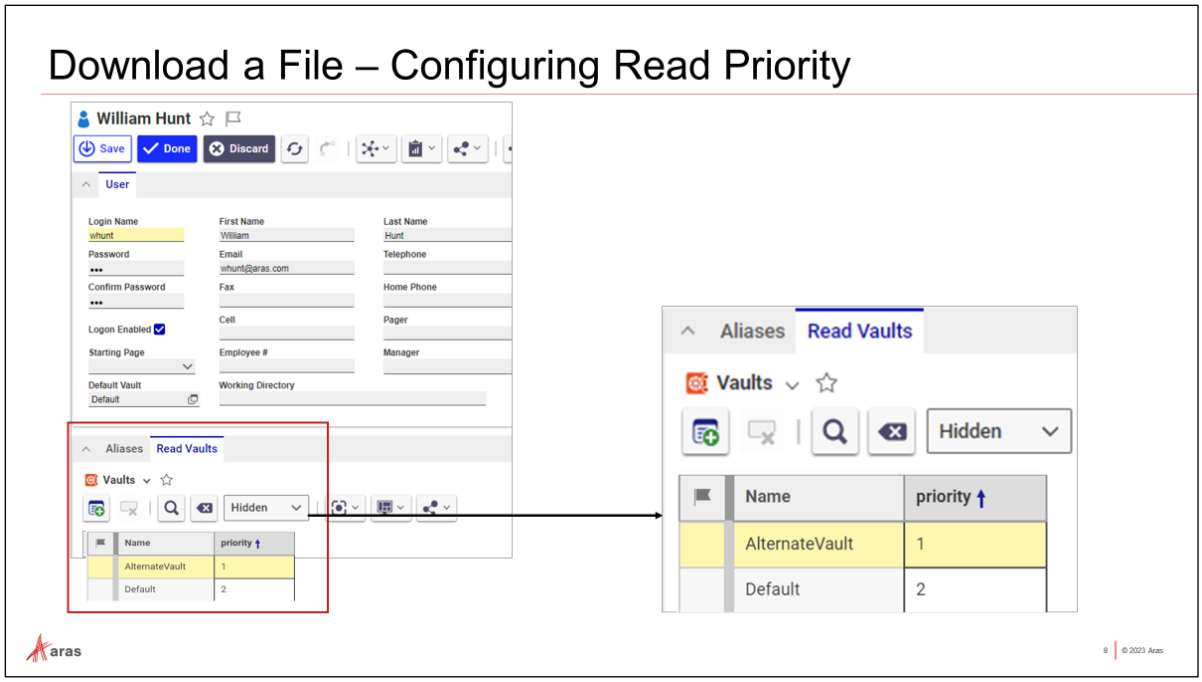
1. View the vaults by navigating to Administration > File Handling > Vaults and running a search.
2. You should see two vaults: **Default** and **AlternateVault**.
3. Open the **Default** vault and explore its settings.
4. Open the **AlternateVault** vault and explore its settings.



Download a File – OnDemand replication rules & Vaults

When a user saves a file, it is always saved into their assigned Default Vault.

If there are no rules that replicate files, Aras Innovator always checks the files table to determine where files are located and delivers the correct version of the file to the user.



Download a File – Configuring Read Priority

By default, when users make a request to download a file, their default vault location is searched first.

Additionally, when a user gets a file from Aras Innovator, the system can check the existence of the file in different vaults, based on the Vaults' Read Priority (if defined).

Setting Read vaults should be done for users to improve performance accessing files, based on latency, bandwidth, server load, and other considerations.

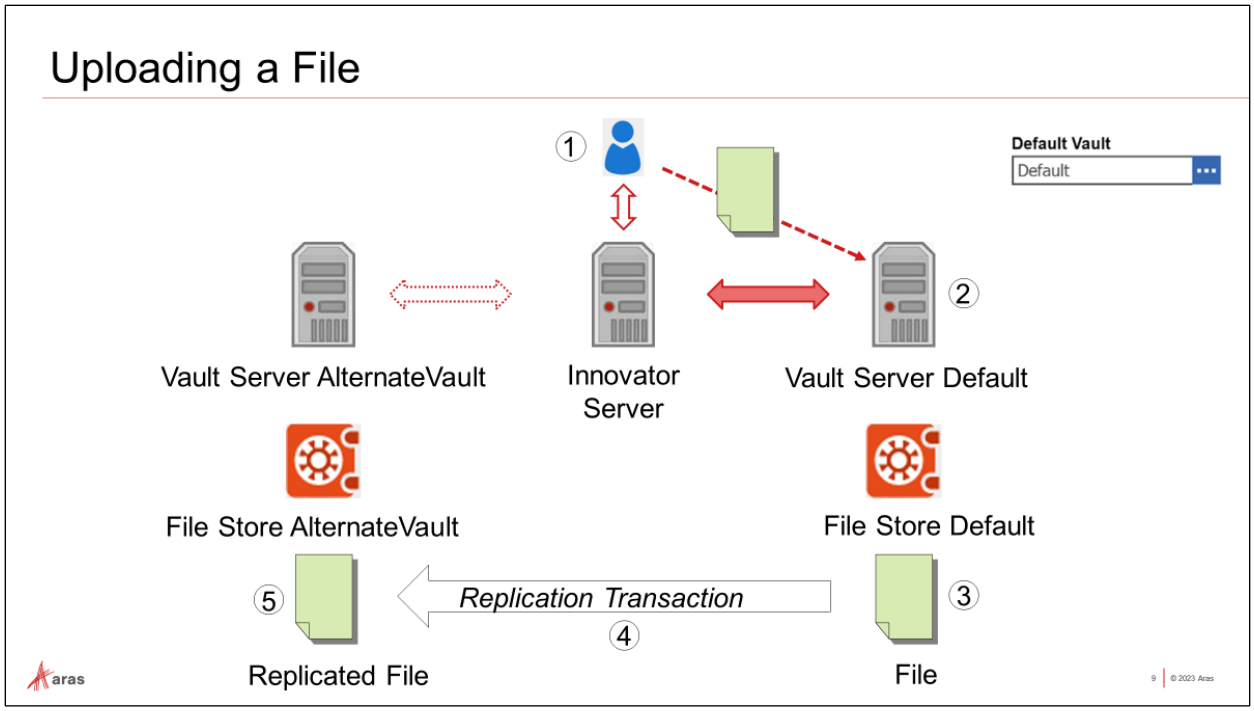
When a user downloads a file, the following locations are checked for the file (in the order listed):

1. The Read vault with the lowest Read Priority number = highest priority
2. Any additional vaults listed on the Read vaults relationship tab in consecutive order
3. The User's default vault
4. Any other vaults in the system

Try it … Configure Read Vault Priority

Terry Adams was assigned to the default vault named **Default**. However, many of the 3D models that he must frequently review reside in an alternate vault named **AlternateVault**. To improve the speed for Terry to access the files, set Read Vault priorities on his user profile.

1. From Administration > Users, open the user *Terry Adams* for editing.
2. Select the Read Vaults tab on the relationship grid and click the Add Vaults button.
3. Choose both existing vaults: **Default** and **AlternateVault**.
4. Set **Default** to Priority *2* and set **AlternateVault** to Priority *1*.
5. Click Done.



Uploading a File

This example shows a process for onChange file replication, in the case of file upload (either a new file or changed file). This example uses two vaults that have been configured to support file replication from the **Default** vault to the **AlternateVault** vault.

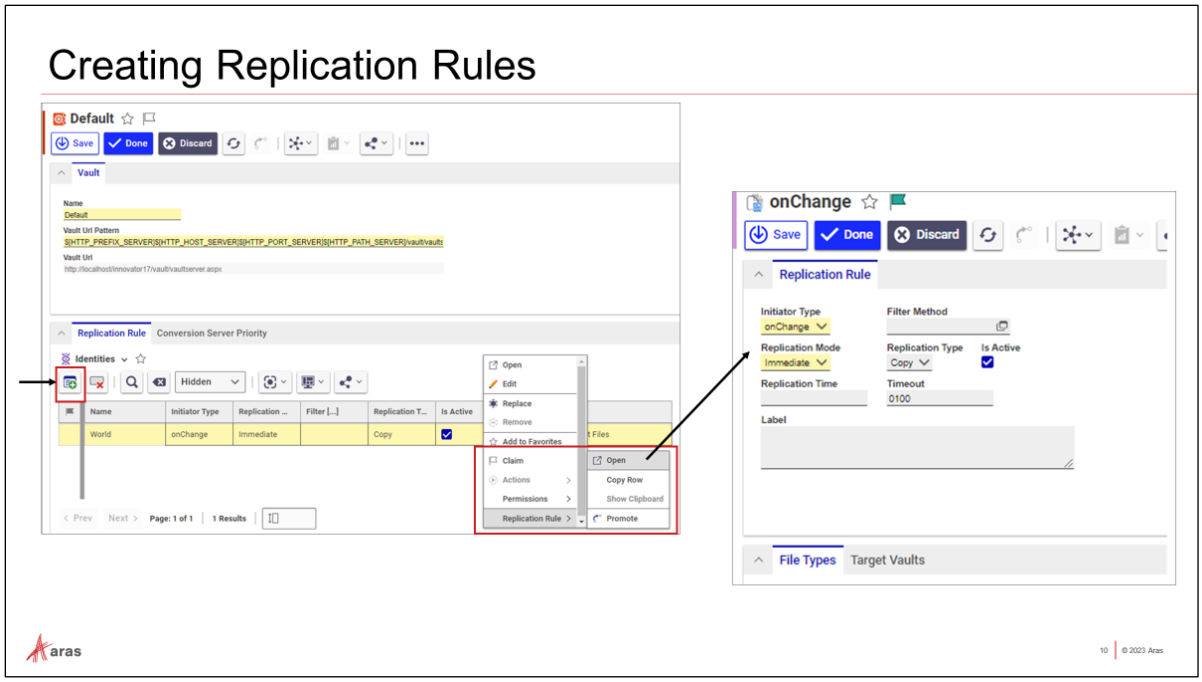
① A User is always assigned to a Default vault in the User profile. In this example, the User is assigned to the default vault named **Default**.

② The Default vault has been configured to copy to the AlternateVault vault when a file is uploaded in the Default vault.

③ The file is first uploaded to the User's Default vault.

④ A Replication Transaction is created to copy the file to the targeted AlternateVault vault.

⑤ The replicated file now resides in both the AlternateVault and Default vaults.



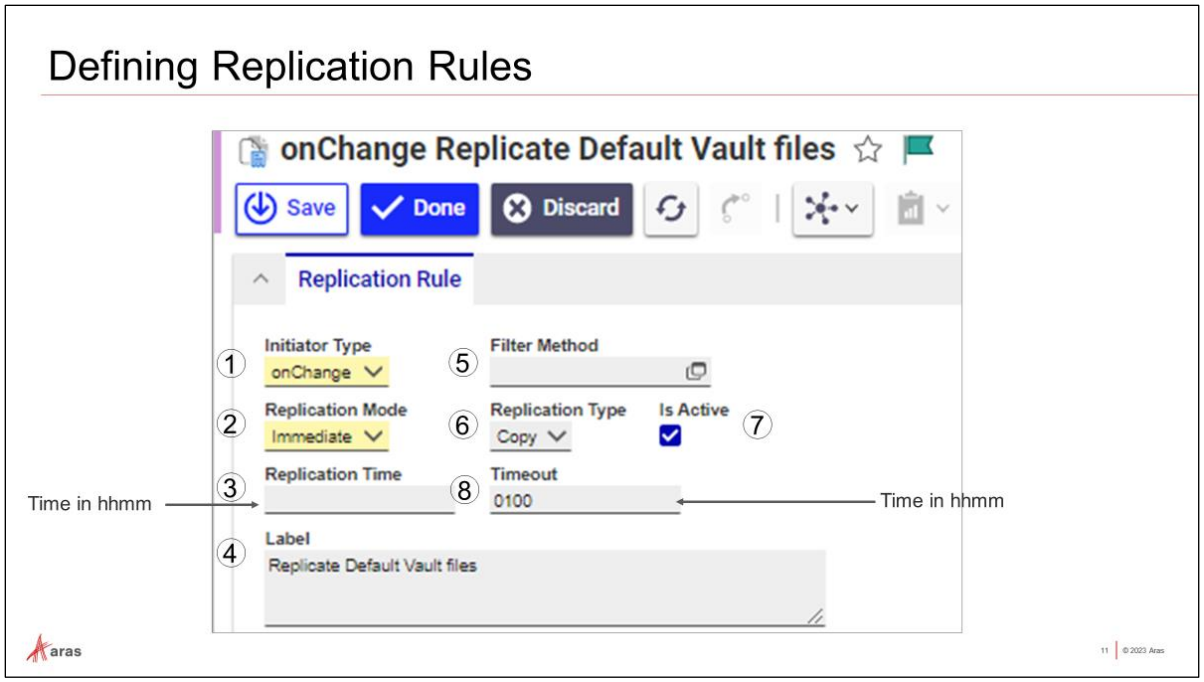
Creating Replication Rules

You define how file replication works in your environment by creating Replication Rules. A Replication Rule is assigned to a vault and associated with an Identity. The Identity defines which Users will have access to the Replication Rule when a replication event occurs. Choose the World identity if you wish the Replication Rule to apply to all users, or you can be more selective by choosing group identities.

The Initiator Type indicates what events will cause a replication transaction to be created. Replication Rules are created on the source vaults. Typically, this would be the vault that was set as the Default vault for a group of users and stores the physical files from an upload.

Try it … Create a Replication Rule

1. Navigate to Administration > File Handling > Vaults.
2. Open the **Default** vault Item for editing.
3. Select the Replication Rule tab in the Relationships grid to create a new rule.
4. Click the Add Identities button and choose **World**.
5. Right-click the Replication Rule row and select Replication Rule > Open.



Defining Replication Rules

Several options are available to indicate when the transaction should be invoked, including immediate execution, delayed, at a scheduled time, or by program invocation.

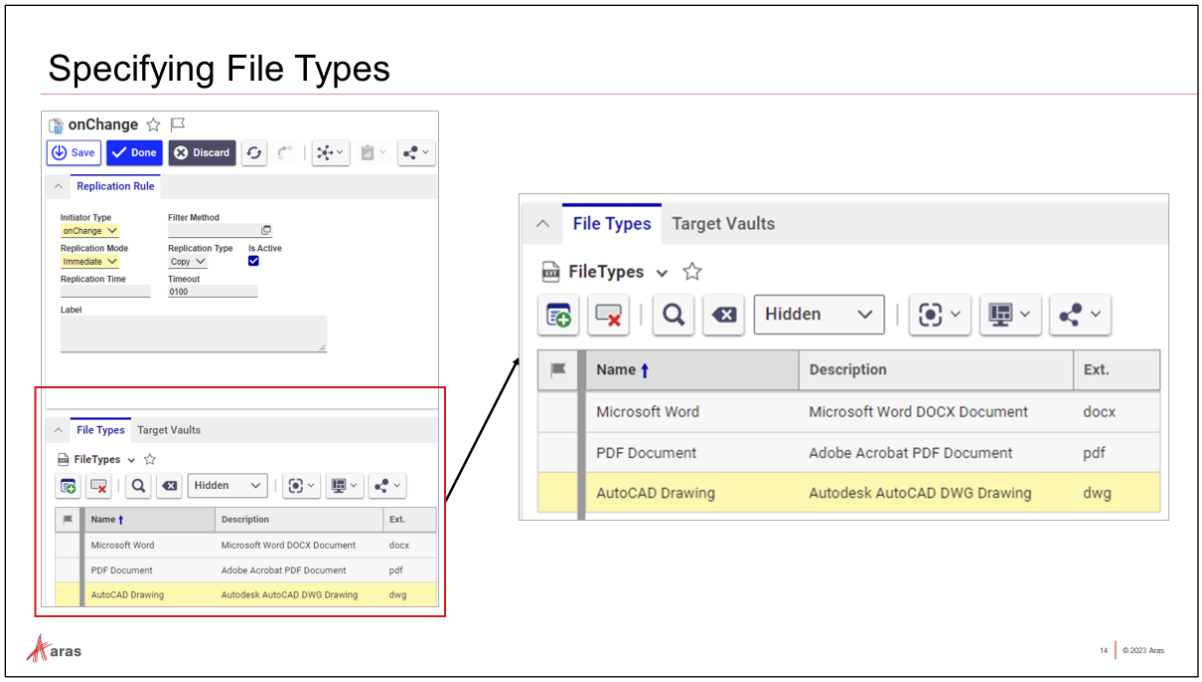
The Replication Rule Form provides the following configuration fields:

①	**Initiator Type**	**onDemand** – Replicates when users belonging to the identity set on the rule have a read vault(s) defined. The target vault will match the highest read priority of the user initiating the request. Transactions will not be initiated if the requested version of files already reside in the vault. Similar OnDemand rules should be created on each vault for consistent behavior. **onChange** – Replicates when an identity has added or changed a file in a vault. **onEvent** – An executed Method, with replicate programming instructions, activates this event.
②	**Replication Mode**	**Immediate** – replication will begin without delay, asynchronously. **Delayed** – delay Replication Transaction using the Replication Time field in hours and minutes (hhmm). **Scheduled** – set the transaction time to the UTC time specified in the Replication Time field in hours and minutes (hhmm). If time has already passed for today, it will execute at that time tomorrow. Max time is 2359. **Manual** – transaction will not be processed until a program method call is made.
③	**Replication Time**	Use in conjunction with Delayed and Scheduled transactions. Format in hours and minutes (hhmm).

④	**Label**	The description of the Replication Rule – this text displays in the Replication Transaction Log.
⑤	**Filter Method**	A custom Method that tests if a file should be replicated (or not) when the Replication Transaction is processed. When the Replication Transaction is invoked, the Method receives the current file id, which can be used to determine if a file should be replicated. Example: files over a certain size or in a certain Lifecycle state are not replicated based on business rules.
⑥	**Replication Type**	A Copy or Move of the replicated file.
⑦	**Is Active**	Allows you to choose when rules are active or ignored. You must deactivate a rule and wait for any transactions to finish before it can be deleted.
⑧	**Timeout**	Maximum time (hhmm) allowed between start and completion of a transaction before it is considered Failed. Default value is 0100 (one hour).

Try it … Define the Replication Rule settings

6. On the Replication Rule, select **onChange** for the Initiator Type field.
7. Select **Immediate** for the Replication Mode field.
8. Choose **Copy** for the Replication Type field.
9. Click to select (enable) the Is Active checkbox and click Save.

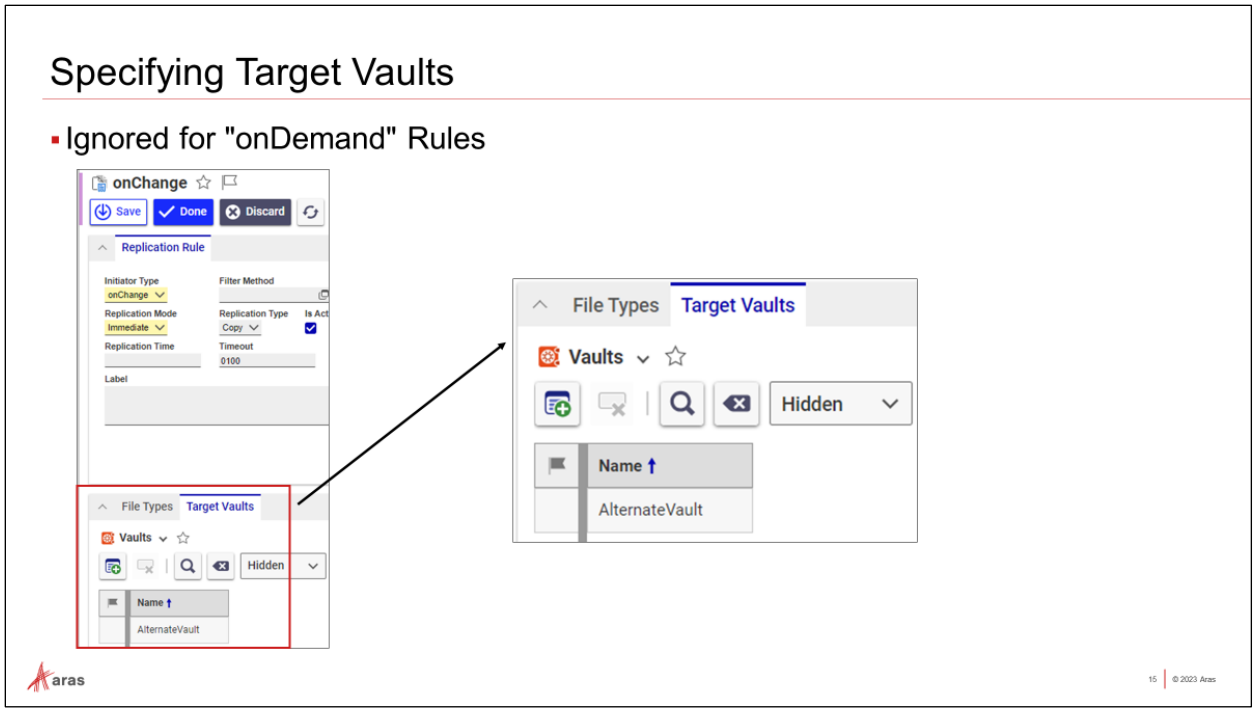


Specifying File Types

To be restrictive, you can specify which file types will be included as part of a Replication Rule transaction. If no types are provided, then all file types are accepted.

Try it … Specify File Types

10. Select the File Types tab on the Relationships grid and click the Add FileTypes button to select from the list of known file types.
11. Select two file types matching sample files that can be used for testing the replication rule (e.g. Microsoft Word, PDF Document).
12. Click the Save button.



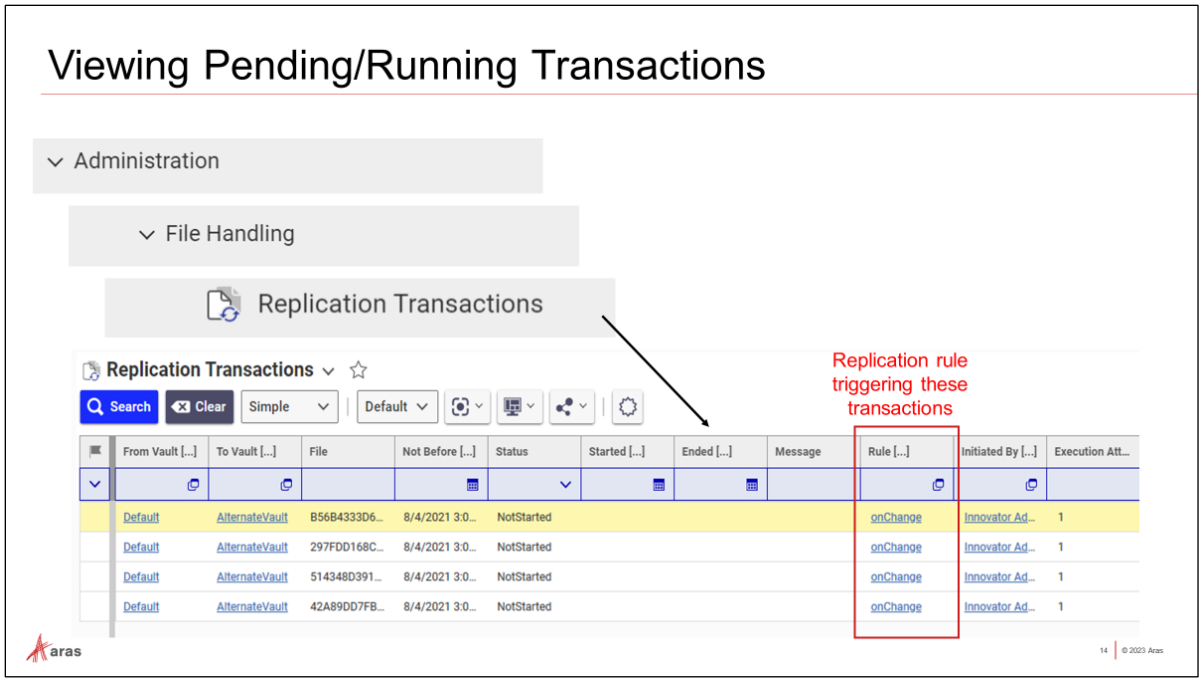
Specifying Target Vaults

Target vaults indicate where the copied (or moved) files are sent. Where the Initiator Type is OnDemand, the target vault is the initiating user's highest 'read' priority vault; therefore, the target vault is ignored.

Only one Target vault should be specified when the Replication Type equals Move. Having more than one target vault results in transaction errors since the file no longer exists in the source vault after the first completed Move transaction.

Try it … Specify a Target Vault

1. Select the Target Vaults tab in the relationships grid.
2. Use the Add Vaults button to select the Target vault to be configured for this rule.
3. Choose the **AlternateVault** vault and click Done on the Replication Rule.
4. Return to the **Default** Vaults tab and click Done.
5. Create a Document Item or any other item, attach two files in it and click Done.



Viewing Pending/Running Transactions

Each Replication Rule can trigger several transactions. For example, for a *OnChange* replication rule, one transaction is created every time a file is uploaded.

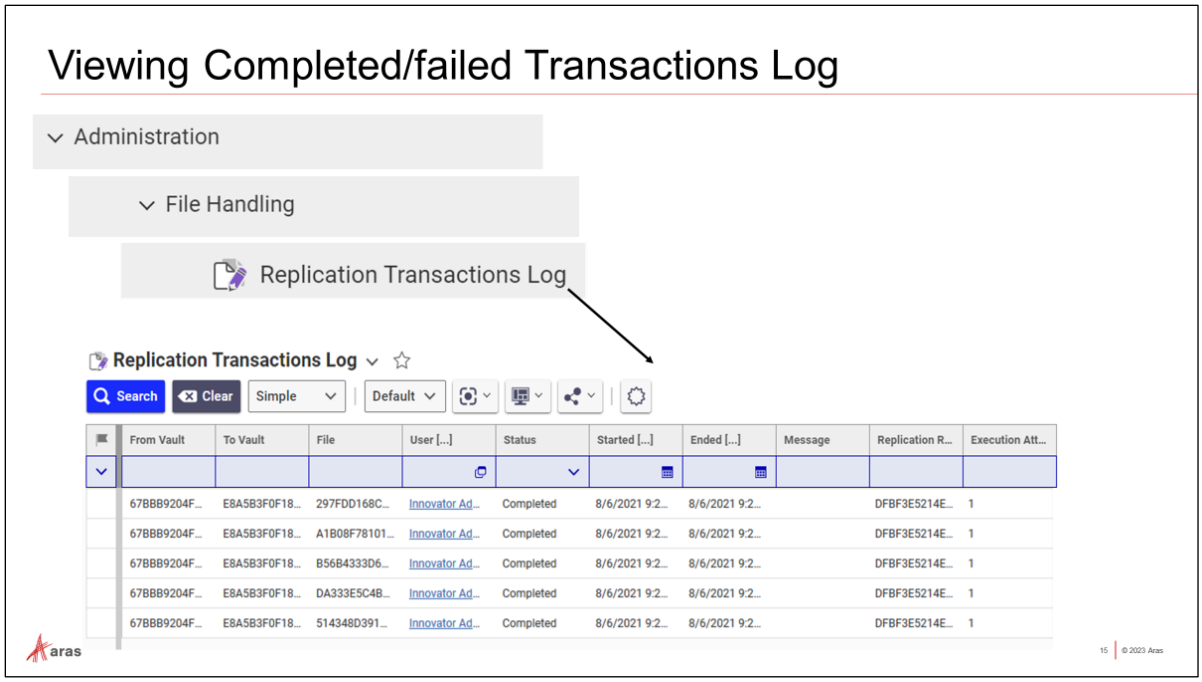
You can display pending or in-process transactions to monitor the pipeline.

The search grid displays the following Properties for review:

- **From Vault** – the Vault that contains the latest version of the file.
- **To Vault** – The Target vault for the Replication Transaction.
- **File** – The id of the file to be replicated.
- **Not Before** – The date and time restriction for delayed or scheduled transactions.
- **Status** – Not Started/Pending/Failed/Completed/Discarded.
- **Started** – The date and time the transaction started.
- **Ended** – The date and time the transaction ended.
- **Message** – The Label text from the Replication Rule.
- **Rule** – The Replication Rule Initiator Type.
- **Initiated By** – The Identity that initiated the transaction.
- **Execution Attempt** – Displays the total number of recorded attempts performed by the system to replicate the file.

Try it … View Transactions in Process

1. Navigate to Administration > File Handling > Replication Transactions.
2. Click Search to see a list of any pending or in-process transactions.

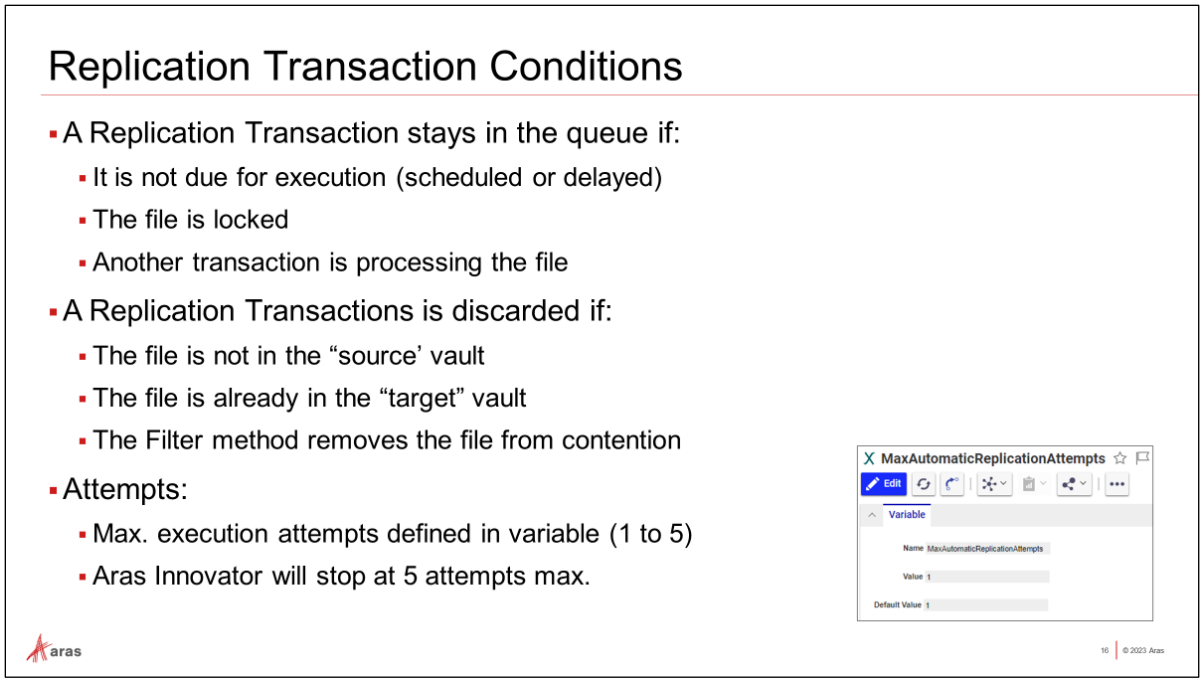


Viewing Completed/failed Transactions Log

A list of failed and completed transactions can be viewed by Administrators, using a second log "Replication Transactions Log". Troubleshooting methods can be used to see if the item containing the file might have been claimed or there was network interference.

Try it … View Completed or Failed Transactions

1. Navigate to Administration > File Handling > Replication Transactions Log.
2. Click Search to see a list of any completed or failed transactions.
3. Check and correct the Replication Rule for any failed transactions.
4. After evaluating and correcting any Replication Rules, a system administrator can use the right mouse button context menu to resubmit any transaction with status = failed.



Replication Transaction Conditions

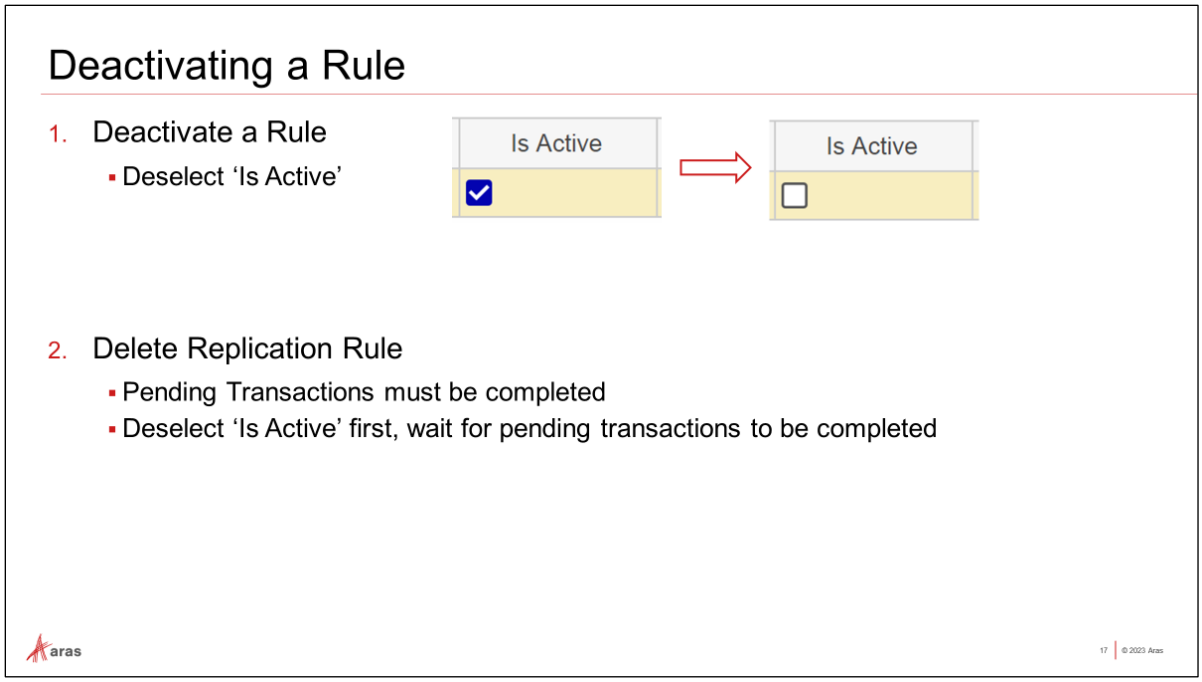
Replication transactions are processed if they meet the conditions specified in the replication rule.

There are several reasons why a file might not be replicated, as shown in the above slide. A *Not Started* transaction remains in the queue for future processing. Discarded transactions are removed from the queue and marked as Discarded in the Transaction Log.

System Administrators can view the logs containing the history of successful and failed replications. They should diagnose and resolve the cause of the failure.

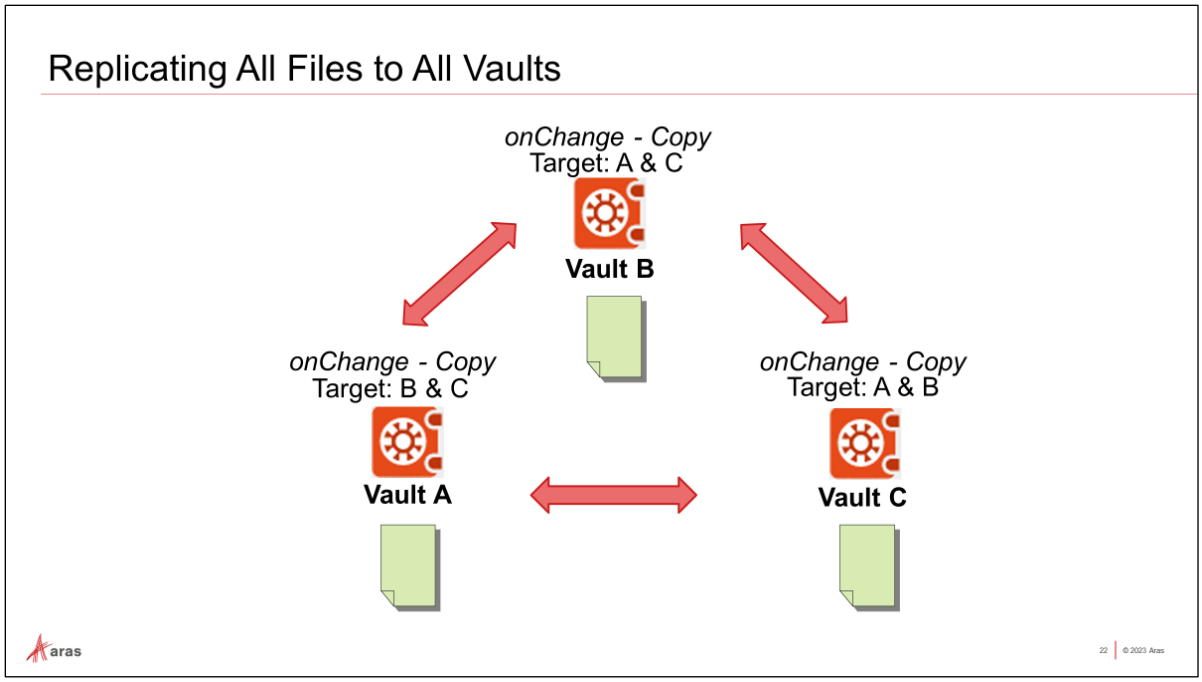
Aras Innovator can be configured to automatically resubmit failed replication transactions up to 5 times. To configure this setting, you must change the value of the **MaxAutomaticReplicationAttempts** Variable Item (integer value between 1 and 5). Aras Innovator will always stop automatic resubmission after 5 attempts.

Administrators can also manually resubmit any **Failed** Replication Transaction Log Item as a new Replication transaction using **Resubmit** from the Action menu for the Replication Transaction Log Item. Replication transactions can be manually resubmitted any number of times until the transaction succeeds.



Deactivating a Rule

You can delete a Replication Rule completely if it is no longer needed. However, if there are any outstanding Replication Transactions, you cannot delete a rule until those transactions are completed. If you wish to delete a rule, you can first set it as Inactive, to inhibit the creation of any new replication transactions, wait for any existing transactions using that rule to complete or manually delete them, then delete the Replication Rule.

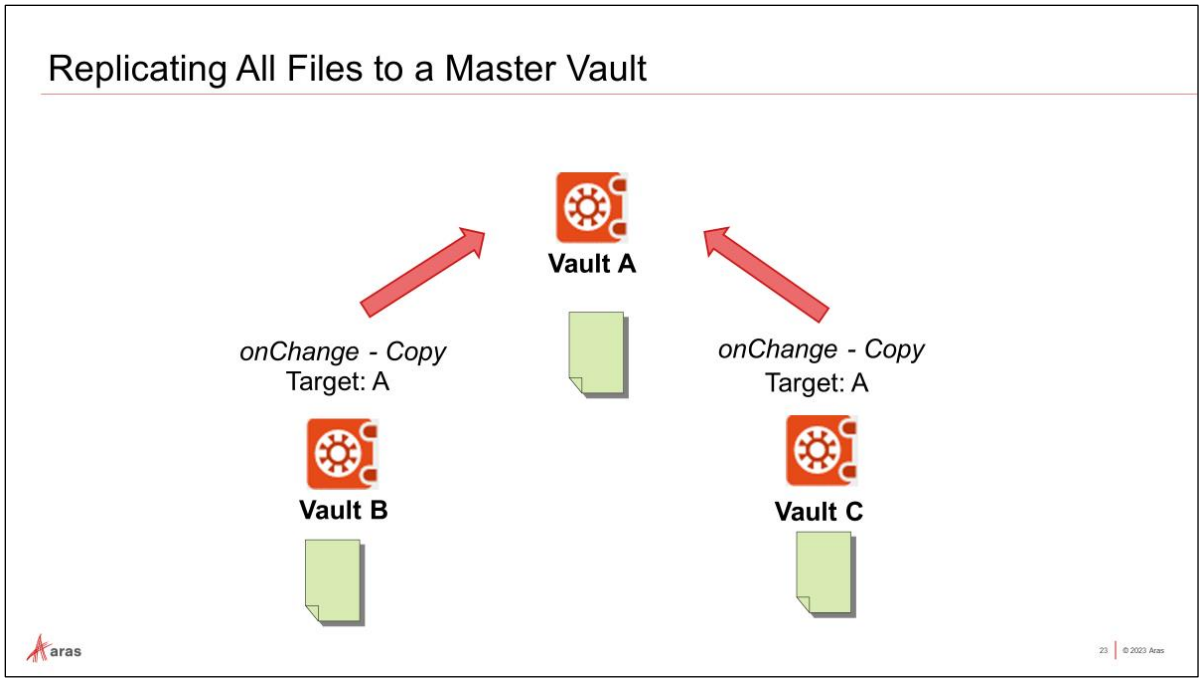


Replicating All Files to All Vaults

This configuration guarantees that there will be redundancy and all files will be stored in all vaults.

An onChange Replication Rule is assigned to each vault in the system. Each Replication Rule targets the other vaults in the system for replication when a file is changed in any vault.

Depending on number of users and the system load, these rules could be scheduled for off hours when network traffic is lighter.



Replicating All Files to a Master Vault

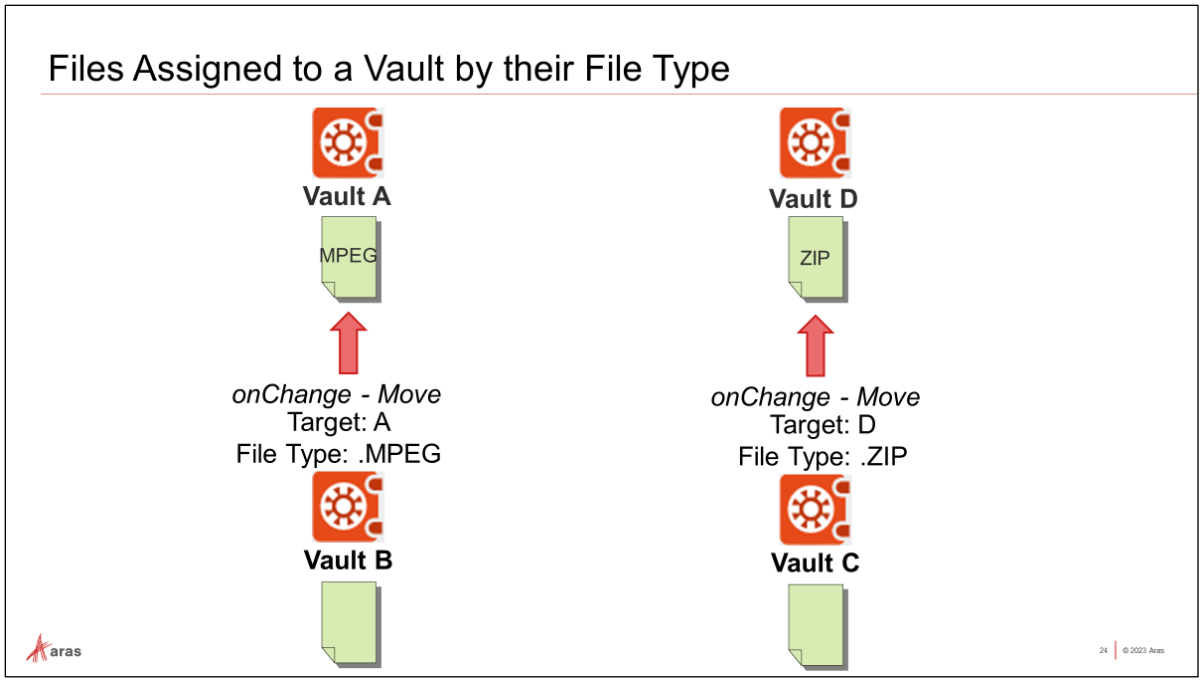
This configuration is good for an organization where there is one main location with a large disk array like a data center, and numerous local vaults at smaller satellite locations.

This configuration is effective when you want to maintain a central repository containing all files and you do not want to assign the master vault as every user's Default vault.

This configuration could be used in combination with onDemand rules to copy files to each user's highest priority Read vault to reduce network latency.

In addition, you may want to set this type of replication to be scheduled during off hours to capitalize on periods with low server use and lower network traffic.

Finally, for this configuration, one Replication Rule must be set on each vault, except the Master vault.

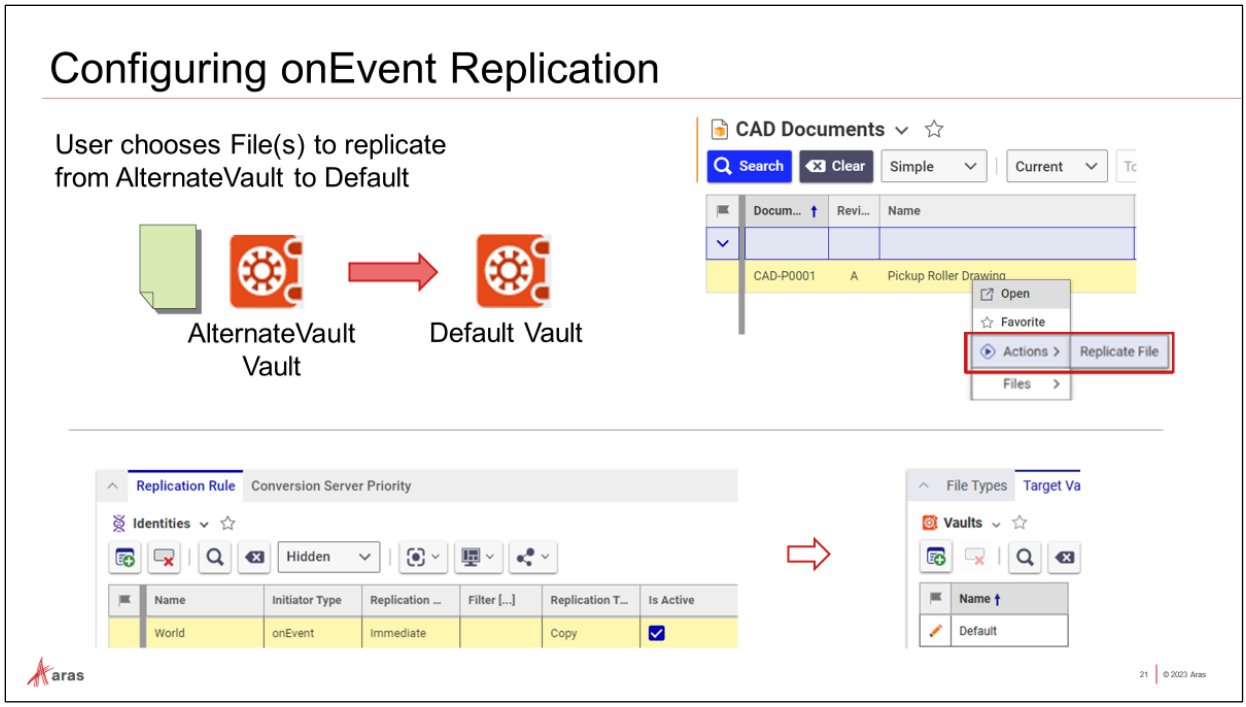


Files Assigned to a Vault by their File Type

This configuration guarantees that files are moved to a vault based on their File Type.

In this configuration you use smaller distributed vault servers, that are not meant to store numerous extremely large files permanently. You then establish rules to move large files to a vault server with a large disk array for long term storage.

As shown above, each User's Default vault (B and C) targets a specific vault based on a File Type. An onChange Replication Rule, with the configured File Type, moves the file from the User's Default vault to the vault specified as the location for that type of file.

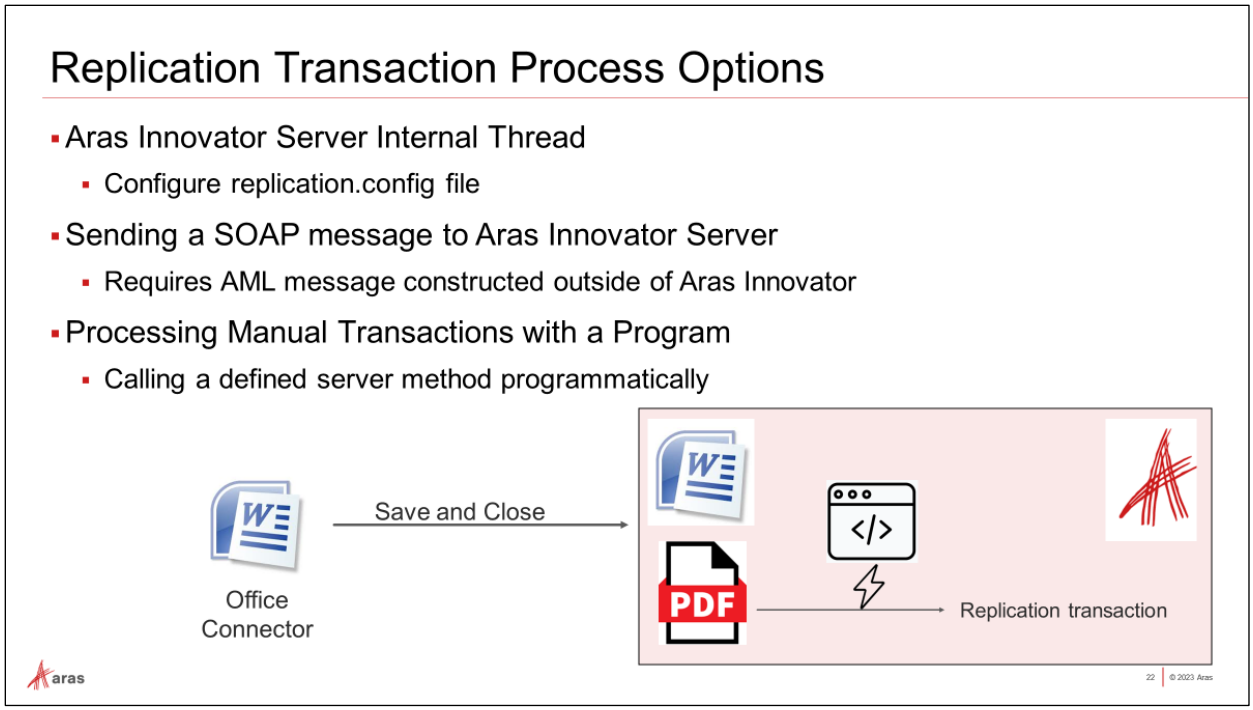


Configuring onEvent Replication

This example demonstrates a User having the ability to select a file for replication using a custom configured Action and Method (note: this action does not exist on training environment). The user selects a file and chooses the Replicate File menu item to invoke a Method call.

The AlternateVault vault containing the file starts a Replication Transaction based on the onEvent Replication Rule and targets the **Default** vault for the copy operation.

This same approach could be used to provide a response to a Lifecycle promotion event to replicate Document Item files to a Target vault(s) when a Document Item reaches a certain State (e.g., Released).



Replication Transaction Process Options

File replication can be invoked either within the Aras Innovator Server as a thread, outside of Aras Innovator as an Aras Innovator service/scheduler, or through a program configured with the standard Windows scheduler.

Internal Thread

You can configure an internal thread to handle the processing of replication transactions that have been created using Replication Rules. A **replication.config** file in the Aras Innovator Server enables file replication using the standard internal thread. The default settings for configuring this file are:

- Replication Interval – 60 seconds
- Username/Password – vadmin/vadmin
- Max Transactions Batch – 25
- Max Transactions Pending – 50
- db – [Aras Innovator database name]

SOAP Message to Aras Innovator Server

The Aras Innovator Server can respond to a specially formatted SOAP message to invoke processing of Replication transactions.

Processing Manual Transactions

You can configure Replication transactions to execute using a manual process, which requires calling a specifically designed server method to process these transactions. In the above example, a method is triggering a transaction when a pdf is created/updated through ‘Save and Close’ in Office Connector.

Summary

In this unit, you learned how to configure file replication. You should now be able to:

- Create Replication Rules
- View Replication Transactions and Logs
- Describe Additional vaults
- Configure a Preferred vault
- Deactivate and Remove Replication Rules

Review Questions

1. Where do all files added or changed by a User get written to first?

2. How do you specify which vaults should receive a copy of a file when it is changed?

3. How would you restrict a replication rule so that only AUTOCAD files were replicated?

4. Where do you specify the Read vault Priority list?

Lab Exercise

Goal

Configure a file replication rule.

Scenario

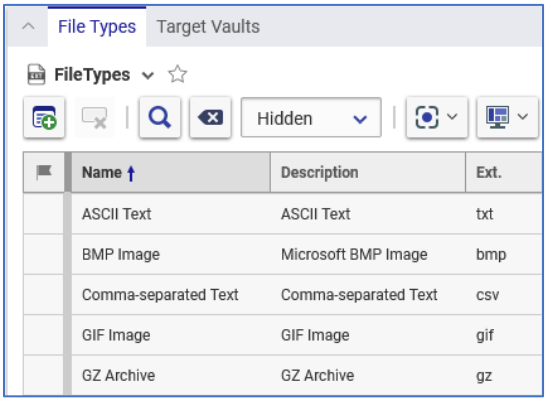
In this exercise, you configure file replication. You define a Replication Rule using the configured Default and AlternateVault vaults.

Steps

1. Make sure your student login (an Administrator) is assigned to the **Default** vault.
2. Ensure that the Read Vault Priority is set as follows:

   Default = 1, AlternateVault = 2

3. Navigate to Administration > File Handling > Vaults and open the **Default** vault Item for editing.
4. Open the Replication Rule tab in the Relationships grid, click the Add Identities button, and choose **your student login (an Administrator)**.
5. With the **Default** vault open for editing, select **onDemand** for the Initiator Type field.
6. Select **Immediate** for the Replication Mode field.
7. Choose **Copy** for the Replication Type field.
8. Click to select (enable) the Is Active checkbox.
9. Click the Save button.
10. Right-click on the **onDemand** Replication Rule row and choose Replication Rule > Open from the context menu.
11. Click the Edit button and from the File Types tab on the relationships grid, add a new relationship (Add FileTypes button) for each file type to be supported, as follows:



12. Click the Save button.
13. Locate the Target vaults tab in the relationships grid and use the Add Vaults button to select the **AlternateVault** vault as the Target vault for this rule.
14. You can click Done for the Replication Rule.
15. Return to the **Default** vault tab and click Done.

Unit 18 Sending Notifications

Overview

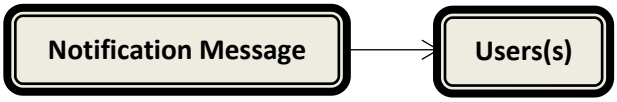
In this unit, you will learn about the two types of Notifications that can be sent from Aras Innovator: User Notifications and e-mail Notifications.

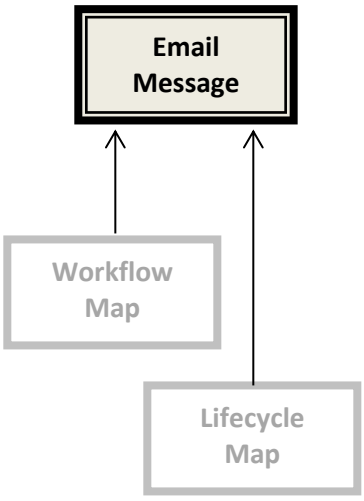
User Notifications appear as alerts to users in an Aras Innovator session. These notifications can be used to alert a user about an upcoming event (e.g., system maintenance, updated feature, etc.).

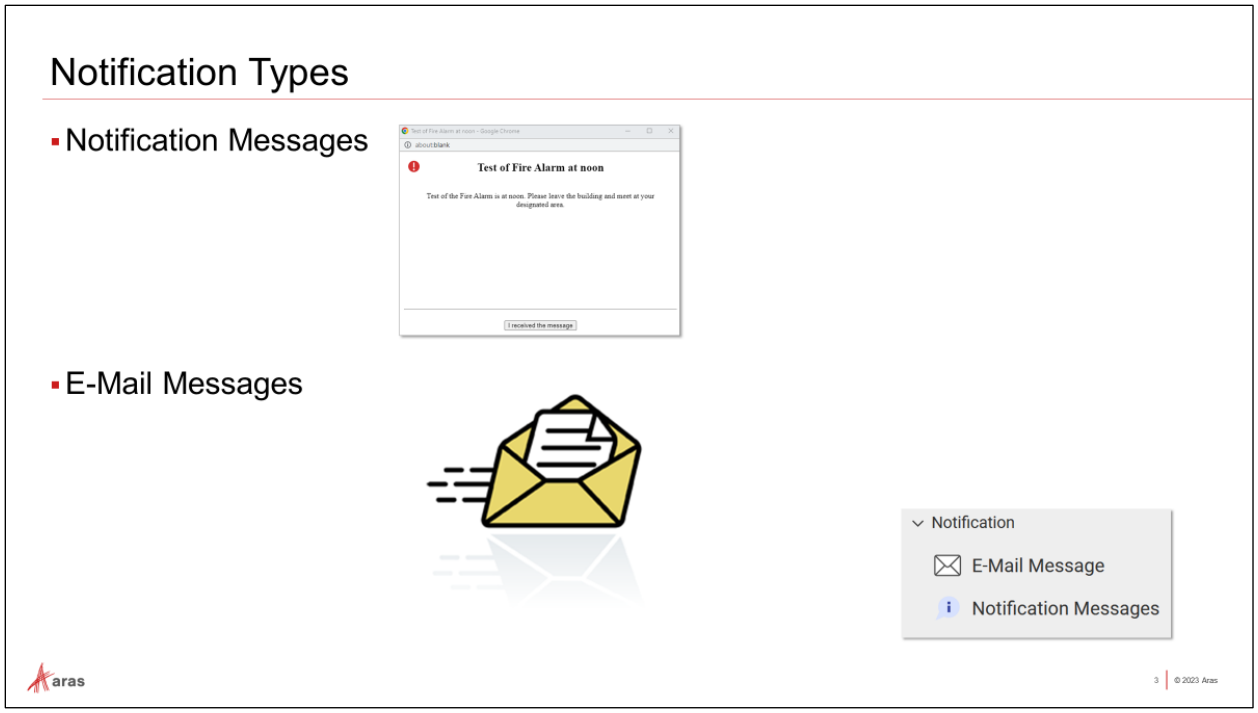
E-mail Notifications are customized e-mail messages that can be sent to users. For example, when an Item is promoted to a new Lifecycle state or on a workflow activity. You will learn how to create e-mail Message items and configure them for a Lifecycle map or Workflow map.

Objectives

- Understand Notification Types
- Explore Notification Messages
- Create E-Mail Messages
- Use Variables in E-Mail Text
- Configure an E-Mail in a Lifecycle map
- Configure an E-Mail in a Workflow







Notification Types

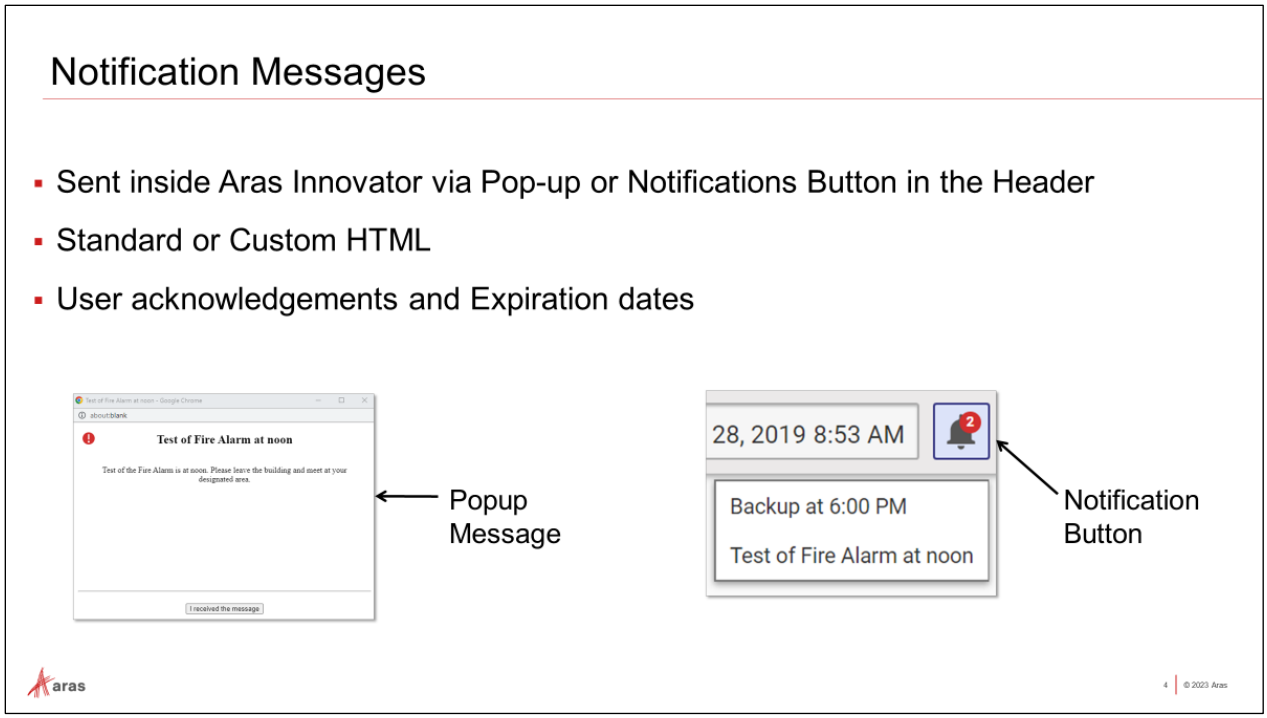
Aras Innovator offers two Notification Types, User Notification Messages and E-Mail Notifications.

Notification Messages

There may be times when you need to notify a group of users while they are working in the Aras Innovator Environment. An administrator can configure a message that will be broadcast to a specific group or alias Identity.

E-mail Messages

You can configure email messages from Aras Innovator that send e-mails to one or more Identities based on an Item Lifecycle state or workflow event.



Notification Messages

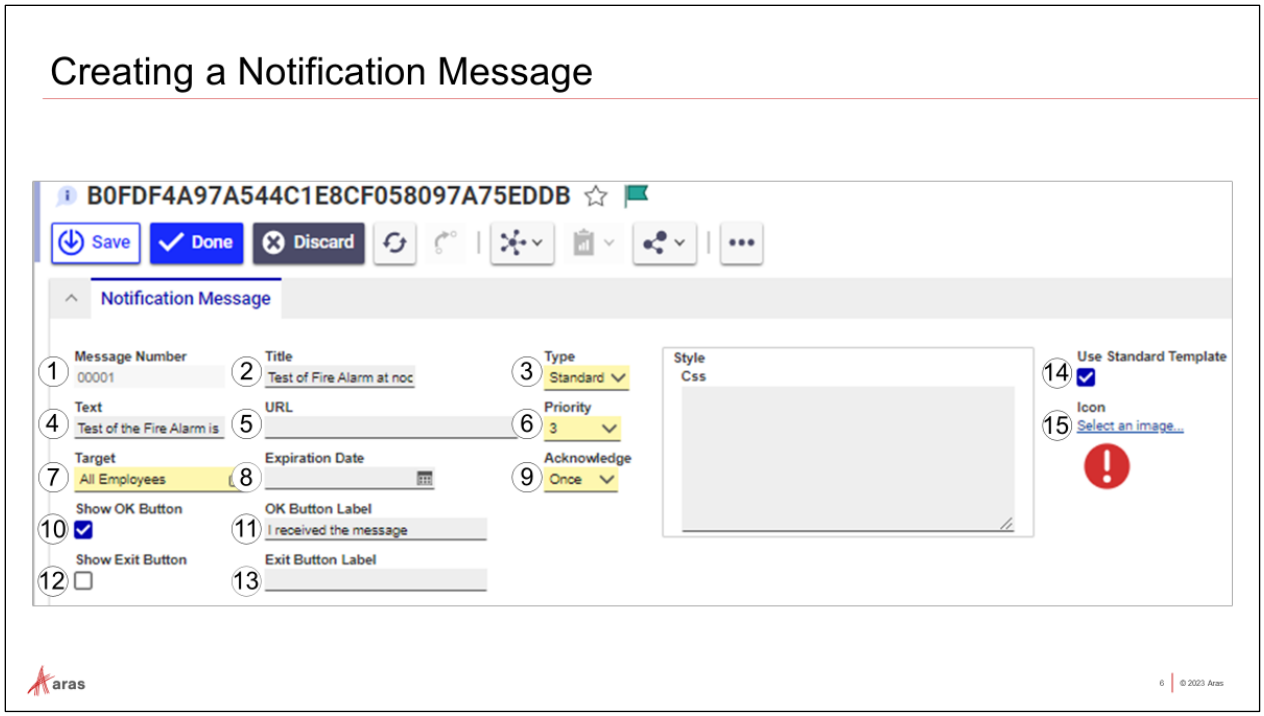
There may be times when you need to notify a group of users while they are working in the Aras Innovator Environment.

You can configure messages to just display in the Header or to pop-up in a separate window. You can also determine if a user must acknowledge a message, which is recorded for review. You can also configure messages to expire after a certain date, which automatically removes them from a user's queue.

Viewing Notification Messages

Notification Messages are indicated and accessed by the Notifications button. You find the Notifications button in the Aras Innovator Header just left of the User Menu button. When a standard message is sent to a user, a message counter increments by 1 to indicate the number of messages waiting to be read.

By clicking the button, the titles of all new messages can be displayed. You can configure the message priority so that more important messages display before others in the list. Clicking on the title opens a window containing the message title and the text as well as optionally displaying the OK and Exit buttons. Messages configured as a pop-up window pop-up automatically.



Creating a Notification Message

The following fields define a message template:

①	**Message Number**	Each message is automatically assigned a sequence number, starting with 00001.
②	**Title**	The title of the message (multilingual strings supported).
③	**Type**	Standard indicates that the message will appear in the Header. Popup indicates the message will only appear as a pop-up window when the message is received.
④	**Text**	The message text (multilingual strings supported).
⑤	**URL**	If supplied, the Title text becomes a hyperlink to the web URL provided.
⑥	**Priority**	Higher priority messages appear above lower ones, ordered by importance.
⑦	**Target**	The Identity that will receive the message – this can be a group Identity.
⑧	**Expiration Date**	The date when the message will expire. Expired messages are removed from each users' queue. If blank, the message will never expire and will only be removed if the administrator deletes the notification message.

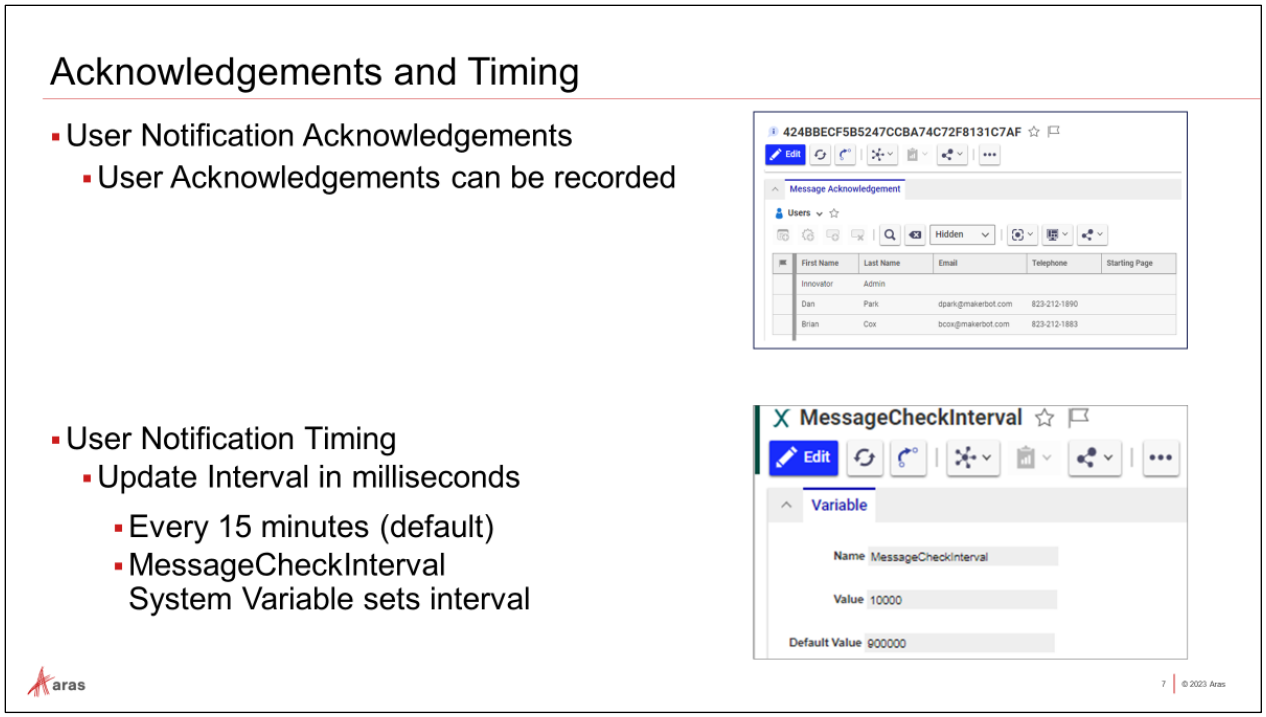
⑨	**Acknowledge**	**Once** Indicates that when users click the OK button on a message it will be removed from their message queue and the event will be recorded for later viewing by an administrator. **Always** indicates the message will always appear in the user's message queue until it expires, or the administrator deletes the notification.
⑩	**Show OK button**	The OK button is considered the acknowledgement button for a message that is configured to be acknowledged once, then the window is closed.
⑪	**OK button Label**	The text for the OK button (supports multilingual string).
⑫	**Show Exit button**	The Exit button closes the window without further activity, which is useful if the user wants to read the message text later.
⑬	**Exit button Label**	The text for the Exit button (supports multilingual string).
⑭	**Use Standard Template**	A standard template is available to quickly create a new message, or you can use your knowledge of HTML to create a completely customized one. When enabled, it uses a default template xml file, located on the Aras Innovator server. Disable for custom HTML templates.
⑮	**Icon**	Select an image to display on the message.

Try it … Create a Notification Message

1. Navigate to Administration > Notification > Notification Messages and create a new Notification Message.
2. Provide the field values as shown in the following table:

Title	*Test of Fire Alarm at noon*
Type	*Standard*
Text	*Test of the Fire Alarm is at noon TODAY!!!*
Priority	*3*
Target	*All Employees*
Acknowledge	*Once*
Show OK Button	*Enabled*
OK Button Label	*I acknowledge*
Show Exit Button	*Enabled*
Exit Button Label	*Remind me later*
Use standard template	*Enabled*
Icon	*Feel free to select an image, if desired*

3. Click Done. Once the Notification Message is saved, it will be sent to currently logged on users based on the next message check interval. The message displays immediately upon login.
4. After the next message check interval (10 seconds in the environment). click the Notifications button and click to open the Fire Alarm notification. If the message check interval is more than 10 seconds, log out and log back in.



Acknowledgements

Administrators can open the Notification Message template to view acknowledgements.

Tip

The Message Acknowledgement Relationship Item stores properties about the acknowledgement. For example, it may be useful to expose the created_on or modified_on Properties in the search grid.

Timing

An administrator may set the polling interval used by the system to check for new notification messages (both pop-up and standard). Any unacknowledged pop-up message will also pop-up again in the next interval, even if user clicked on pop-up's Exit button.

On the training environment, the Message Check Interval *Value* is set to 10000 milliseconds (10 seconds). The *Default Value* is set to 900000 milliseconds (15 minutes).

E-Mail Messages

- Allow E-Mails to be generated from:
  - Lifecycle Transitions
  - Lifecycle States
  - Workflow Activities
  - Visual Collaboration
- Sent to external E-Mail

aras

8 | © 2023 Aras

E-mail Notifications

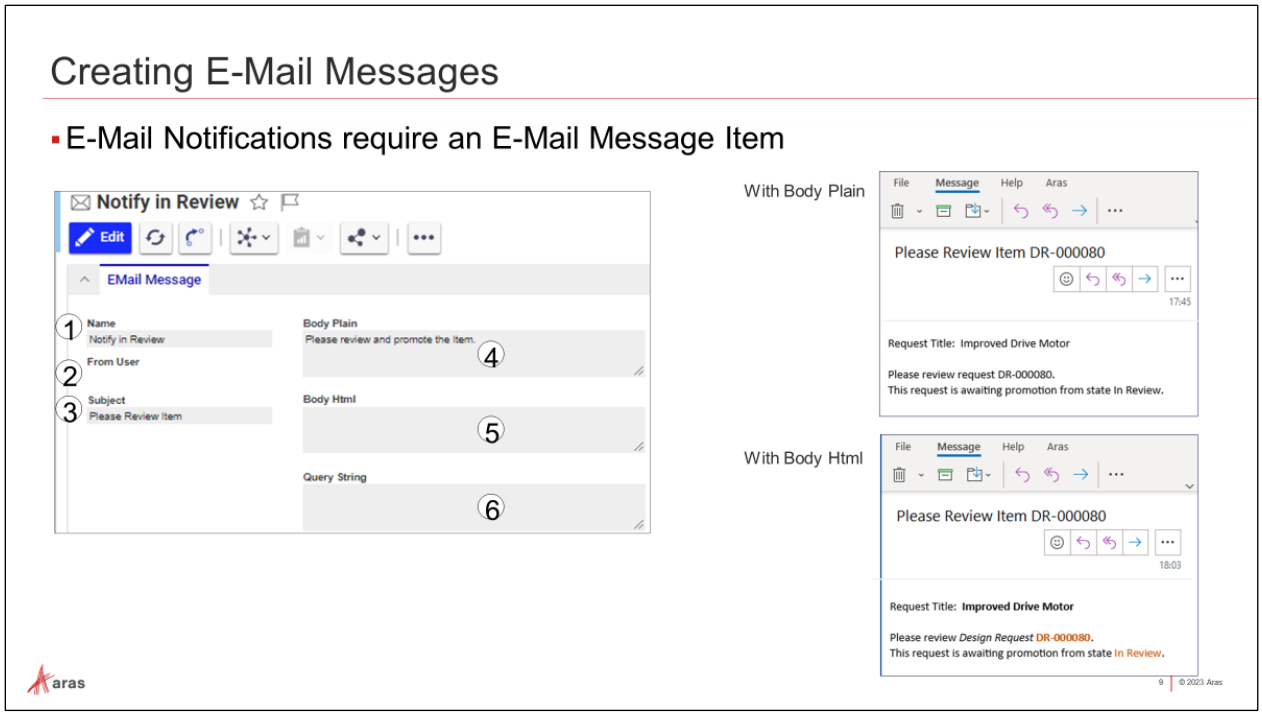
To use e-mail, your Aras Innovator server must be configured to work with a valid SMTP e-mail server. The default configuration uses the IIS mail queue (which must be enabled) but you can also define a specific server. See the *Aras Innovator - Installation Guide* for more information.

If an e-mail server is not available (for testing purposes) you can enable the e-mail debug option. Locate the Aras InnovatorServerConfig.xml file in the Aras server installation directory and remove the disabled setting from the operating parameter of the e-mail debug option:

<operating_parameter key="email_debug_option" value="file"/>

Text files representing the generated e-mails will now be created in the directory associated with the *temp_folder* operating parameter instead of being sent from the SMTP server.

In a Lifecycle map, use e-mail rather than notification messages to notify selected identities that an item has either entered a specific Lifecycle state, or that it is in transition between states. In a workflow, it is also preferred to use e-mails to notify selected identities that a workflow event has occurred. Common workflow events include an Activity becoming active, when an assignment reaches an InBasket, or when an Activity is delegated or escalated to another identity.



Creating E-Mail Messages

To send an e-mail notification, you must first create an E-mail Message Item that contains the text of the e-mail you wish to send, as well as the sender's e-mail information. The e-mail message contains the following fields:

① **Name**

The name of the e-mail message.

② **From User**

Indicates who this e-mail will be from. Choose an existing User or leave blank. Leaving this field blank means the system will use the currently logged on user as the sender of this message. Note that all Users must have a valid e-mail address.

③ **Subject**

The subject of the e-mail Message.

④ **Body Plain**

To send a plain text message, provide the e-mail Message text here.

⑤ **Body HTML**

To send e-mail with embedded HTML tags in the text message, provide appropriate text here.

⑥ **Query String**

An XML representation of the current Item is always available to allow you to use variables in e-mail text. In certain circumstances, you may also want to retrieve other information to use in an e-mail, which requires the use of Aras Adaptive Markup Language (AML).

Using Variables in E-Mail Text – Life Cycle

- You can use variables in a message generated from a LifeCycle Map using the following format:

  ${*property_name*}

  where *property_name* is a valid Property contained in the current Item

- Example:

```
Please review item:
${_item_number}
```

aras

13 | © 2023 Aras

Using Variables in E-Mail– Lifecycle Maps

E-Mails should be instructive to the receivers. Therefore, it is of interest to place relevant Item data into e-Mail messages. You can use specially formatted variables in the Subject and the Body Text of an E-Mail Message.

The Properties of the current Item associated with a Lifecycle map are available for use in sent messages when a Lifecycle State is reached, or when an Item is transitioned from one State to another.

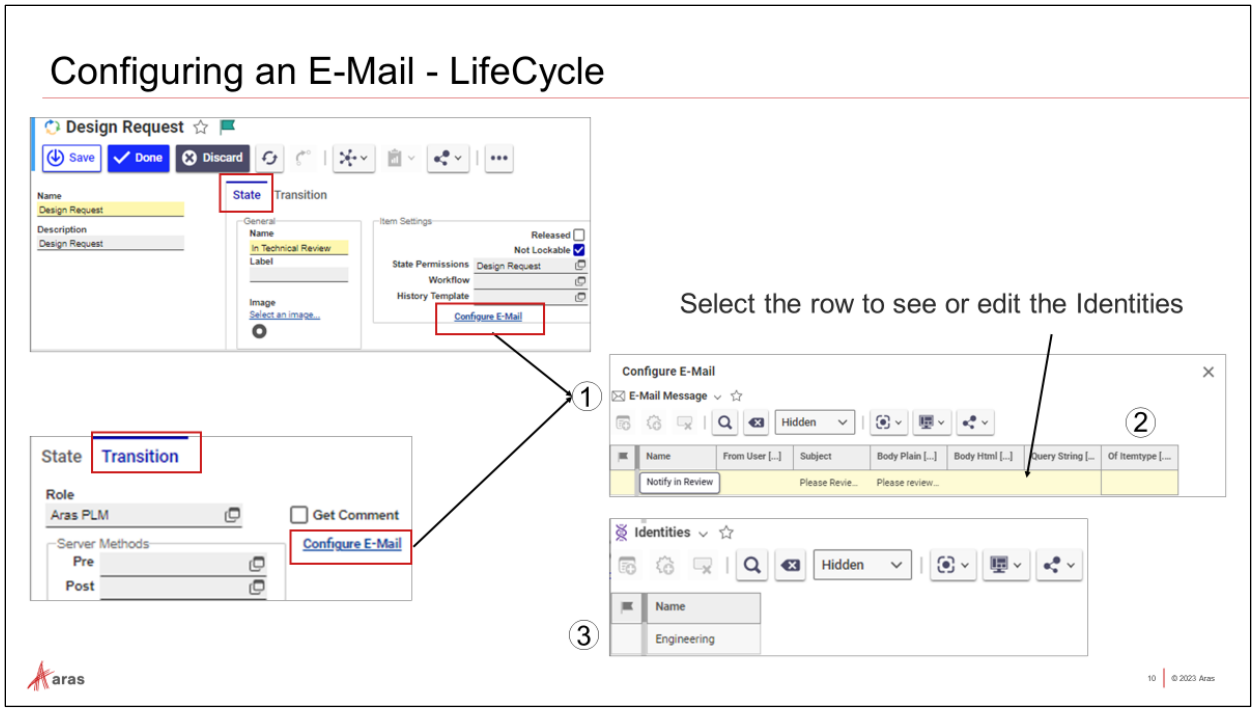
To access a property, use the syntax ${property_name} – where the property_name identifies a valid property defined in the ItemType of the Item. In the example above, the Request Number is substituted in the e-mail message when it is generated.

Try it … Create an E-Mail Message with variables - Lifecycle

1. Navigate to Administration > Notification > E-Mail Message and create a new Email Message.
2. Create a message with the following settings:

Name	*Notify in Review – with Variables*
From User	*(none)*
Subject	*Please Review Item ${_item_number}*
Body Plain	*Request Title: ${_title}.* *Please review request ${_item_number}.* *This request is awaiting promotion.*

3. Click Done.



Configuring an E-Mail - Lifecycle Map

If you configure an e-Mail Message on a State or a Transition of a Lifecycle map, an e-Mail is generated when the system enters the state or passes through the transition.

The Configure E-Mail dialog window displays fields you can configure to send an E-Mail Message Item.

① **Name**

The name of the E-Mail Message.

② **Of ItemType**

If provided, this field limits the use of this E-Mail Message only to the specified ItemType. Note that Lifecycle maps can be shared among ItemTypes. If that is the case, and the e-Mail applies only to a specific ItemType, provide the ItemType name. Leaving this field blank means it works with any ItemType that is used with this Lifecycle map.

③ **Name**

The destination Identity for this message. If a group identity is provided, all members of the identity will receive the e-Mail Message.

After filling in the required values, you simply close the dialog. When you save the Lifecycle map, the e-Mail configuration is saved as well.

Try it … Configure a Lifecycle E-Mail Message

In this exercise, you configure a Design Request Lifecycle State to send the e-mail message you just created.

1. Navigate to Administration > Lifecycle Maps and open the **Design Request** Lifecycle map for editing.
2. On the **In Technical Review** State, click on the Configure E-Mail hyperlink to open the Configure E-Mail window.
3. In the Configure E-Mail window, click the Add E-mail Message button.
4. In the Select Items window, run a search and choose the **Notify in Review – with Variables** e-mail message.
5. In the Relationship grid, use the Add Identities button to choose the **Engineering** Identity.
6. Close the Configure E-Mail window.
7. Click Done.

Using Variables in E-Mail Text – Workflow Notification

- You can provide variables in a message generated from a Workflow Notification using XPath notation.
- Current Item Property:

  ${Item[@type="*ItemType*"]/*property_name*}

- Example:

```
You have been assigned Design Request:
${Item[@type="Design Request"]/_item_number}
```

aras

14 | © 2023 Aras

Using Variables in E-Mail Text - Workflow Notification

Workflow notifications also support the use of variables to retrieve information from the current Item and display it in an e-mail message. Workflow notifications support a more robust syntax using the XPath standard to locate information in the internal Item representation (by AML). In the example above, the Request Number Property of the Design Request Item displays in an e-mail message.

Note

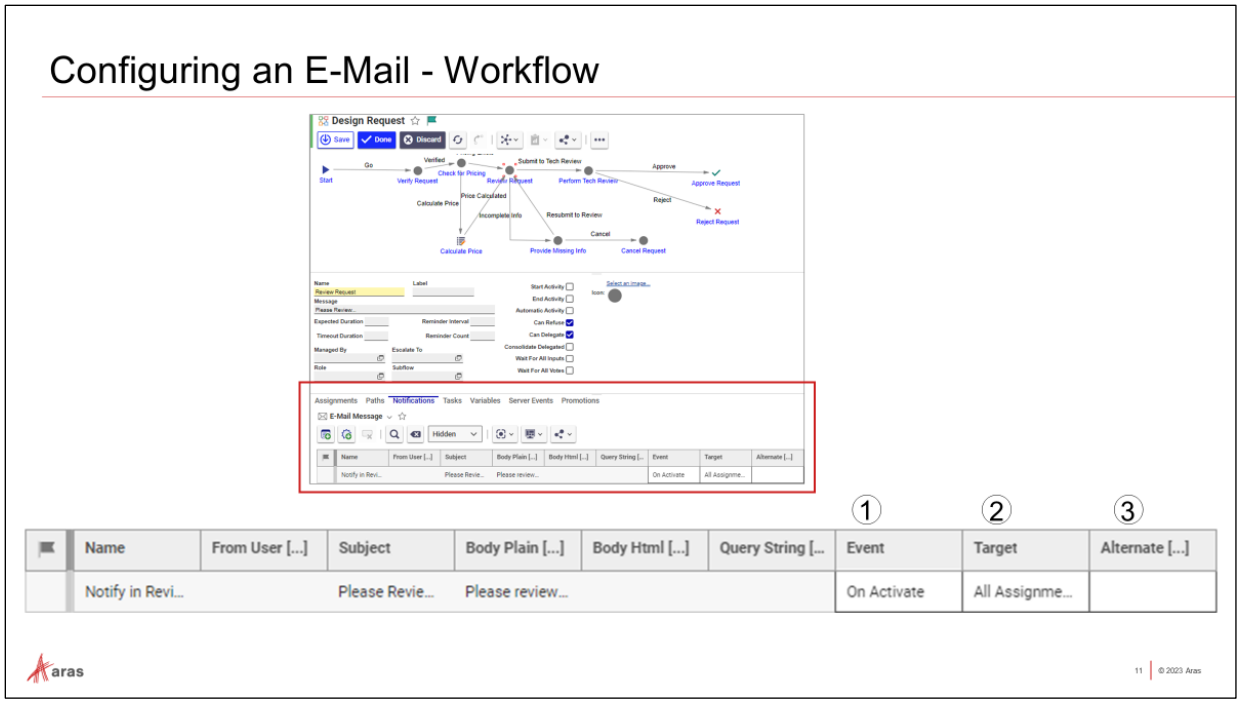
A full discussion of XPath is outside the scope of this class, but you can learn about the XPath standard from many sources on the web.

Try it … Use Variables in a Workflow Notification

1. Create a new E-Mail Message and provide the following values:

Name	*Request WorkFlow Review – with Variables*
From User	*(none)*
Subject	*Request ${Item[@type="Design Request"]/_item_number} waiting to be reviewed*
Body Plain	*Please check your InBasket - Design Request ${Item[@type="Design Request"]/_item_number} needs to be reviewed.*

2. Click Done.



Configuring an E-Mail - Workflow

E-Mail Messages are configured in Workflow Maps on the Notifications Relationship tab. Workflow Notifications can be defined for each workflow Activity. When you select an Activity and view the Notifications tab in the relationships grid, you can provide the following:

① **Event**

The following events are available to indicate when an e-mail should be sent:

- On Activate – when the workflow reaches this Activity.
- On Assign – when an assignment is generated for this Activity.
- On Refuse – when the assignment is refused by a user.
- On Delegate – when a user delegates an assignment to someone else.
- On Vote – when a user votes on an assignment.
- On Remind – when an overdue reminder is triggered based on the duration settings.
- On Due – when the Activity is due to be processed based on duration settings.
- On Escalate – when the Activity is escalated to another identity.
- On Close – when the workflow exits this Activity.

② **Target**

All Assignments - All assignees listed in the Assignments tab for this activity. If there is an assignee, which is a group Identity, all members of that Identity will receive this notification.

Open Assignments - All assignees that have not completed the Activity. Typically, this Target is used with the On Remind or On Due event.

Closed Assignments - All assignees who have completed the Activity. Typically, this Target would be used with the On Close event.

From Identity - Used only for escalation or delegation, this Identity is the one from which the assignment is being delegated or escalated.

To Identity - Used only for escalation or delegation, this Identity is the one being delegated to or escalated to.

Alternate - the Identity defined in the Alternate field.

③ **Alternate**

If the Alternate Target is chosen above, this value is the Identity to receive the message.

Note

There are four system variables available that can be used for any e-mail message.

Variable	Definition
$[USER]	The current logon name
$[ALIAS]	The alias identity of the user
$[DATE]	String containing today's date
$[TIME]	String containing the current time

Try it … Configure a Workflow Notification

In this example, a simple e-Mail Message is configured to notify a user to review a Design Request. It is configured on the Review Request Activity of the Design Request Workflow.

1. Navigate to Administration > Workflow Maps and open the **Design Request** Workflow map for editing.
2. Click to select the **Review Request** Activity, then click the Notifications tab in the Relationships grid.
3. Using the Add E-Mail message button, search for and choose the ***Request WorkFlow Review – with Variables*** message.
4. In the new Relationship grid row, click in the Event field and choose On Activate.
5. In the same Relationship grid row, click in the Target field and choose All Assignments.
6. Click Done for the Design Request Workflow Map.

Summary

In this unit, you learned about User Notifications and E-mail Notifications.

You should now be able to:

- Describe Notifications Messages
- Create an E-mail Message Item
- Use Variables in E-mail Text
- Send an E-mail Message when an Item is Promoted
- Send an E-mail Message using a Workflow Notification.

Review Questions

1. Describe the difference between a Notification Message and an E-Mail Message. Specifically, where does the user see each notification?

2. Where does an administrator configure a Notification Message?

3. What are the four places from which you can initiate an E-Mail Message?

4. What is required before you can configure an E-Mail Message ?

5. What Item values are available to display in an EMail Message?

6. How can you notify a user that an assignment has been delegated to him/her?

Lab Exercise

Goal

Be able to send an e-mail notification from a Lifecycle map and a Workflow.

Scenario

In this exercise, you create notifications for the Change Request to alert users about Lifecycle state changes and active assignments.

Steps

1. The Creator should receive an e-mail message that resembles the following when a Change Request is created. Configure the email message on the **Preliminary** (starting) Lifecycle state.

   Name: *Change Request Created*

Subject: *Change Request ${_item_number} created*

   Message (Body Plain):
   Change Request ${_item_number} has been created.
   Reason: Printer assembly needs to be replaced.
   Cost in USD: ${ _cost_estimate}
   Please submit for review.
   Identity: *Engineering*

2. The **Review Request** Activity of the Change Request Workflow should send the following message to the **CRB** group.

   Name: *Change Request Review*

   Subject:

   Change Request ${Item[@type="Change Request"]/_item_number} needs review.

   Message: *An assignment has been placed in your InBasket.*

   Please review and approve Change Request ${Item[@type="Change Request"]/_item_number}.

3. On the Change Request Workflow's Notification Relationship tab, choose the event of: **On Activate** and choose **All Assignments** for the Target field.
4. As an **Engineering** identity, create a Change Request, select Engineering as the Owner, and process it through its workflow to test your work, keeping in mind the following Activity identities:
   a. The **Engineering** identity you used as the owner, will need to Confirm the Request and Submit it.
   b. Change Specialist I (Terry Adams) will need to Validate the request and a member of Engineering will need to Verify the request.
   c. Verify request goes through the Quality Control Subflow (Administrators).
   d. Finally Review Request to Approve or Reject is by the CRB group identity (Ann Reddy, Dan Park, Diane Prescott).

This page intentionally left blank

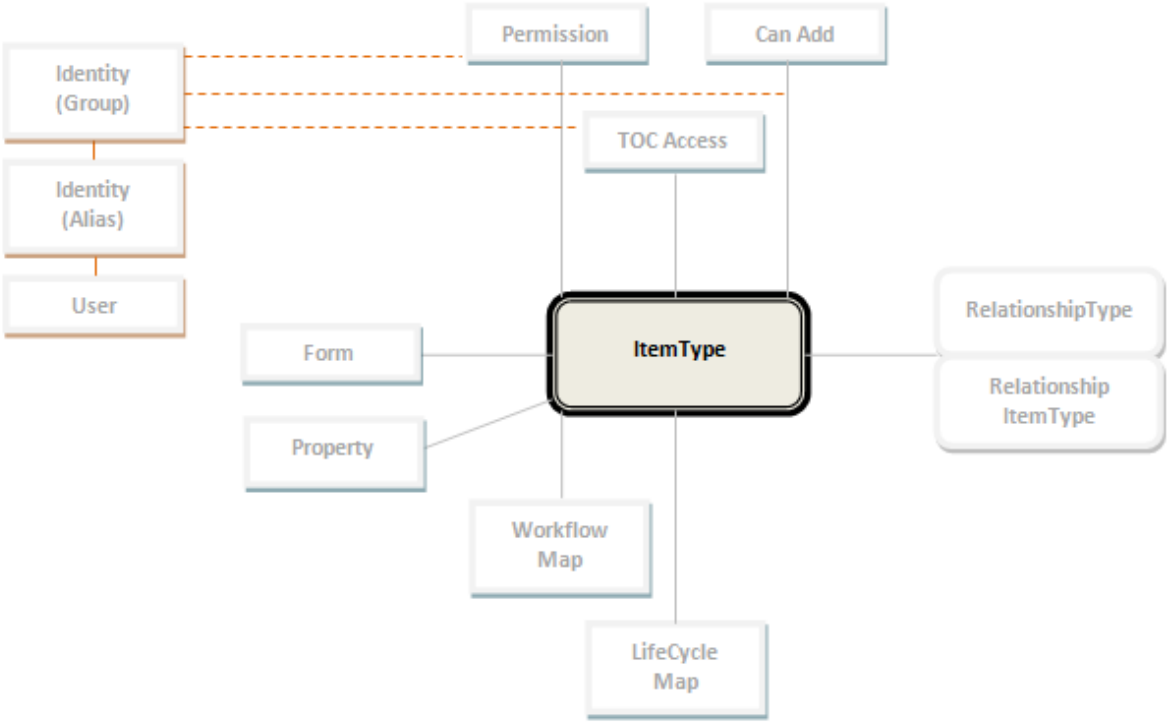
Unit 19 Implementing Poly Items

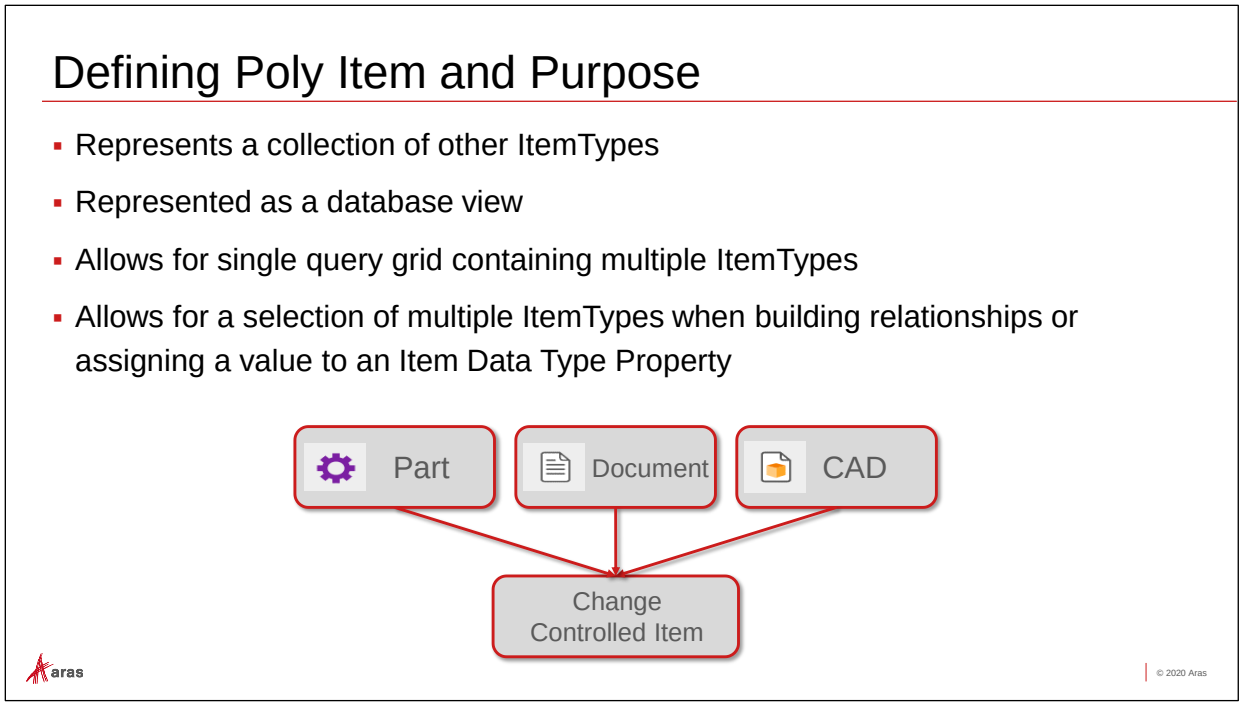
Overview

In this unit, you learn how to define a Poly Item ItemType. A Poly Item is a special type of item that is derived from two or more Poly Source ItemTypes. Poly Items are useful if you need to view multiple ItemTypes in a single search grid (database view). Poly Items are also useful if the related Item of a relationship can be more than one kind of item.

Objectives

- Define the purpose of a Poly Item
- Review an existing Poly Item
- Create a Poly Item ItemType
- Establish a Relationship to a Poly Item
- Use a Poly Item as Item Data Type Property





Defining Poly Item and Purpose

A Poly Item can represent a collection of other ItemTypes. A Poly Item is typically useful in the following use cases:

- When similar properties are created in more than one ItemType and a single view is required to search for an Item.
- When the Related Item of a Relationship can be represented by more than one ItemType.
- When an Item Data Type Property may be represented by more than one ItemType

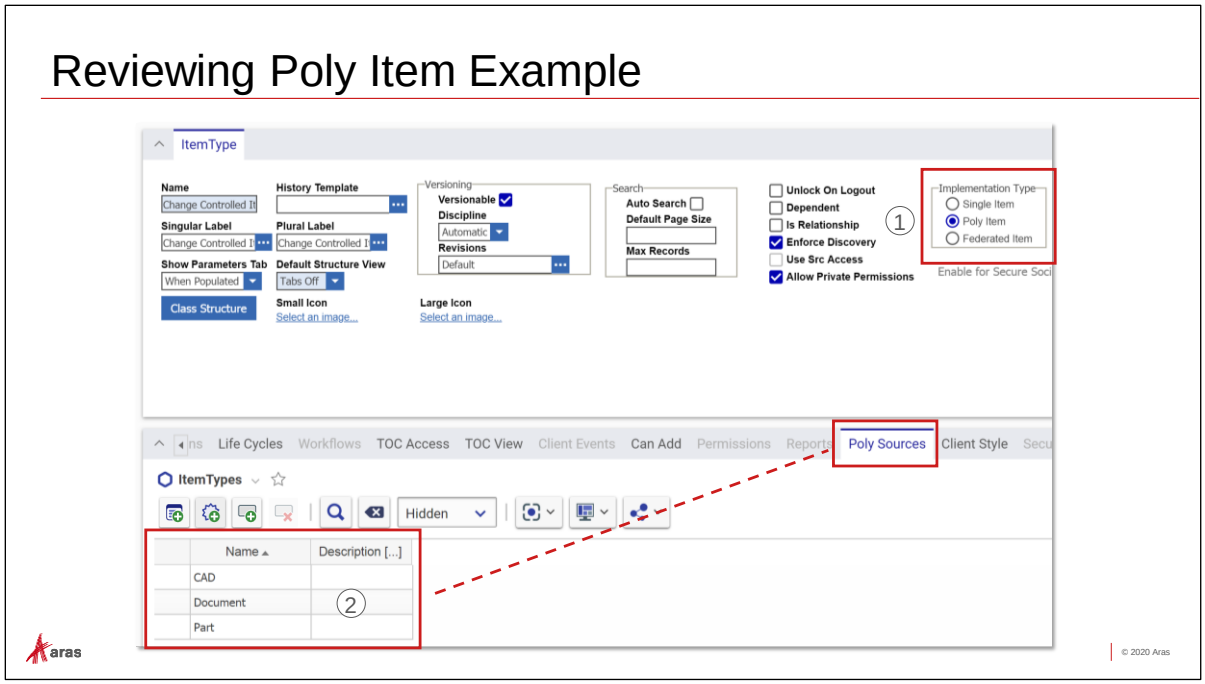
For example, the Product Management solution makes use of a “Deliverable” Poly Item that may be a Document, a CAD Document or some other kind of reference item.

Defining Poly Items

A Poly Item is defined by creating an ItemType marked as a Poly Item and is comprised of two or more Poly Sources.

In the example above, the Change Controlled Item is a Poly Item that can represent either a Part, a Document, or a CAD Document Item.

When an ItemType is defined as Poly Item, a SQL Server database view is created which represents its sources. Once an ItemType has been designated as a Poly Item, it is always a Poly Item.



Reviewing Poly Item Example

A Poly Item is defined by an ItemType that uses the implementation type of Poly Item.

The Aras Product Engineering application uses a “Change Controlled Item” to represent either a Part, a Document, or a CAD Document that is reviewed and modified under change control.

① **Implementation Type**

Poly Item indicates this ItemType will represent two or more ItemTypes that are already defined in the database.

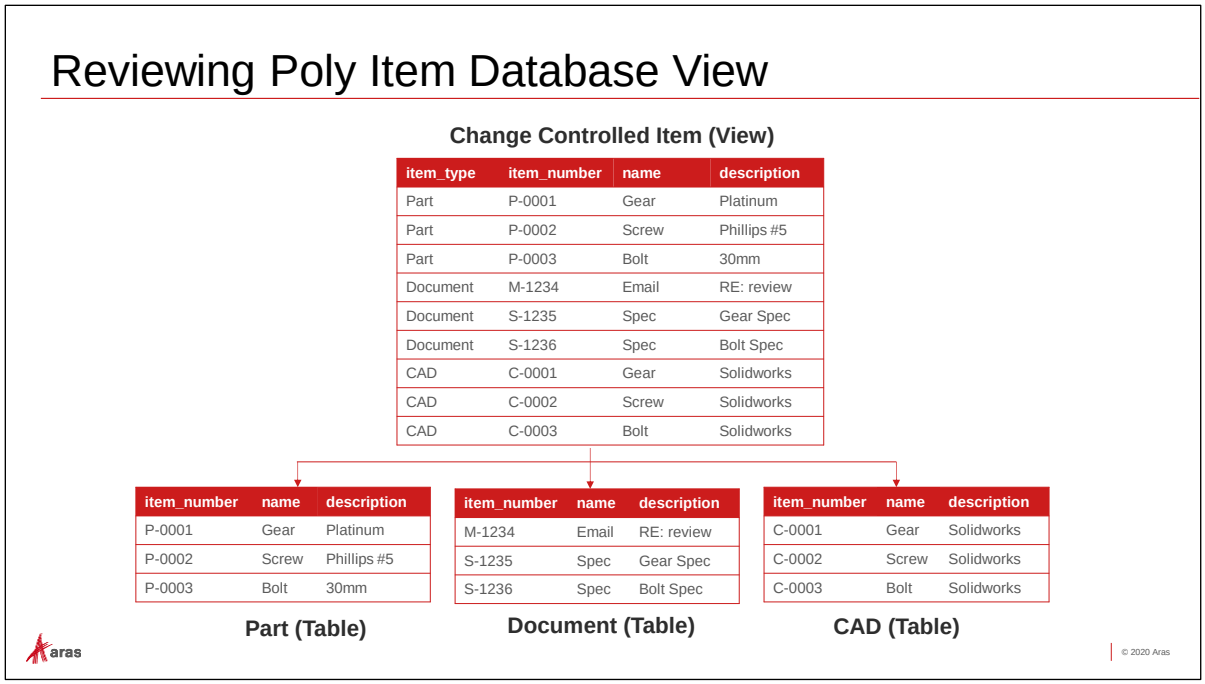
② **Poly Sources**

The “parent” ItemTypes that will define the resulting Poly Items.

A Poly Item ItemType does not implement all the features of a Single implementation ItemType.

Try it … View a Poly Item ItemType

1. Locate the ItemType named **Change Controlled Item** and open it for viewing.
2. Click the Poly Sources relationship tab to see the parent ItemTypes.
3. Close the **Change Controlled Item** tab.



Reviewing Poly Item Database View

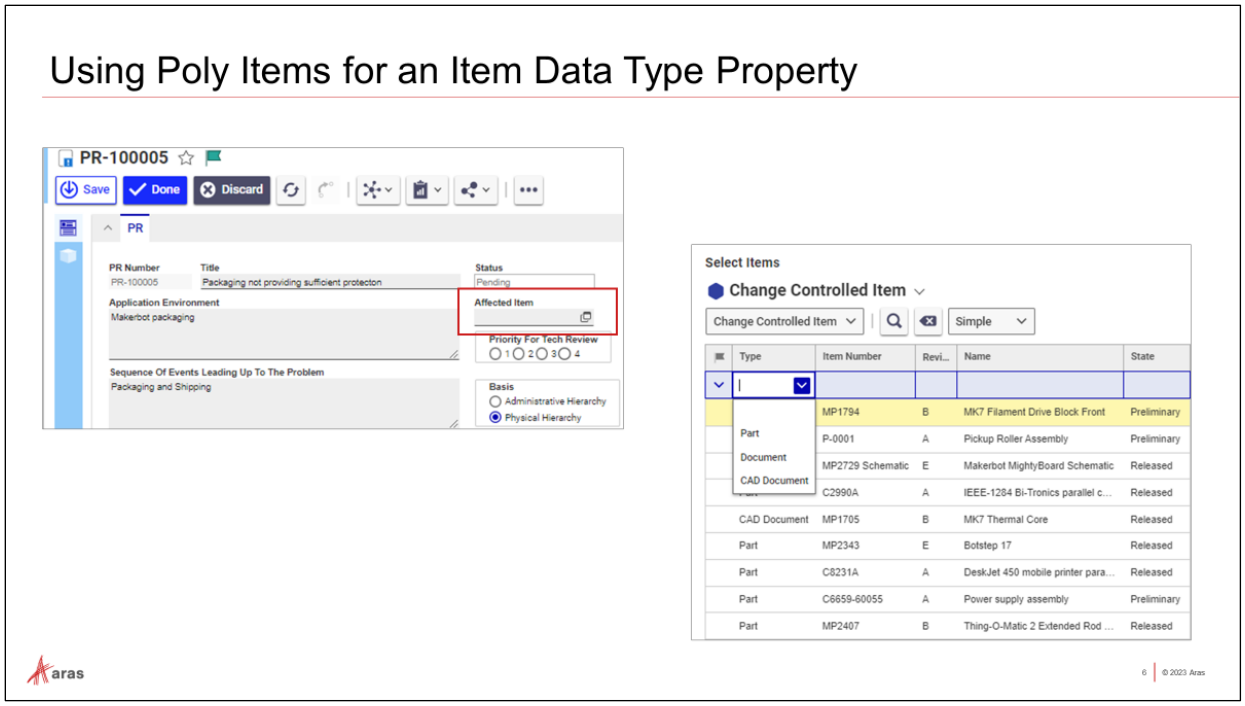
When a Poly Item ItemType is established in the database, a new View is created representing the defined Poly Sources. This enables Aras Innovator to display similar properties in the same main search grid as well as the relationship grid and search dialog windows.

In the example above, the Change Controlled Item View is created in the database to represent the three Poly Sources defined in the Poly Item ItemType.

You can configure which properties display from each Poly Source by defining them in the Properties grid of the Poly Item ItemType and unchecking the Hidden (or Hidden2) checkbox.

Note

If you configure a Property for a Poly Item ItemType, a Property with the same name and a compatible Data Type must exist in all Poly Source ItemTypes.



Using Poly Items for an Item Data Type Property

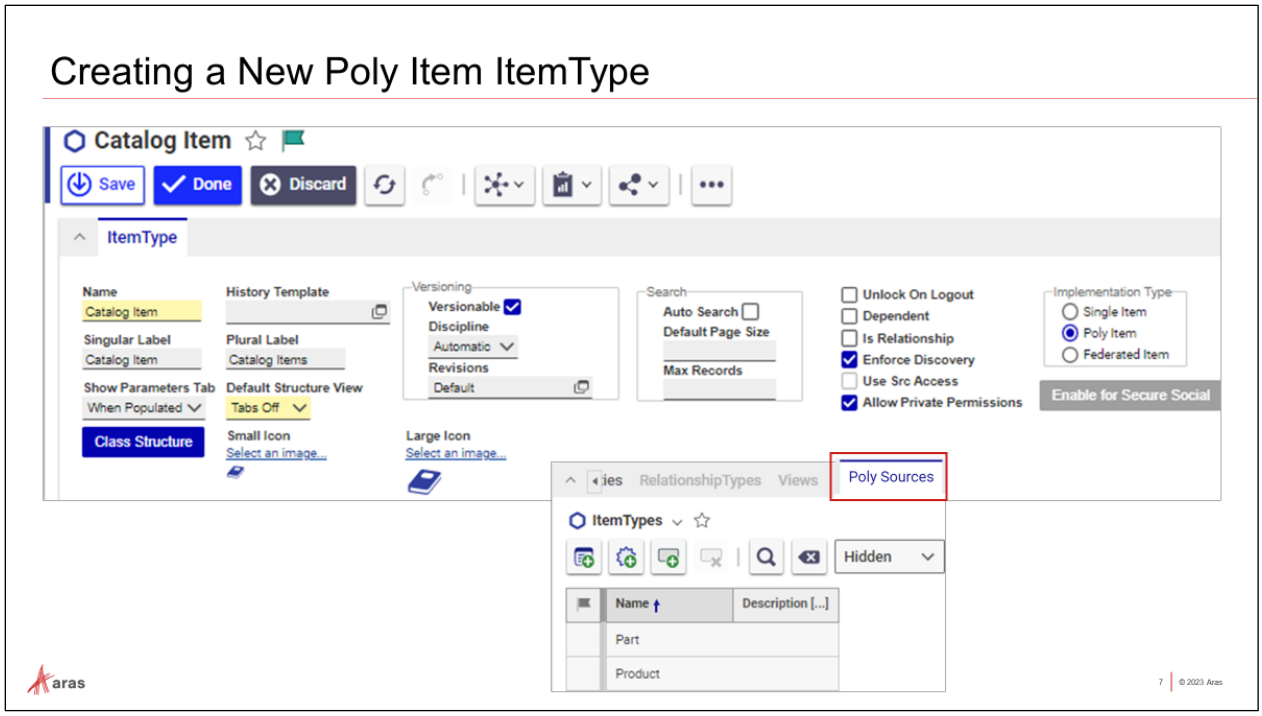
A Poly Item is also useful when the related item of a relationship (or the destination item of an Item Data Type property) can be represented by more than one kind of item.

In the PR example above, the Affected Item Property may be defined as either a Part, a Document, or a CAD Document.

The Property is configured to use the Change Controlled Item Poly Item ItemType. When users select an Item to define a link, they can choose from any of Poly Sources defined in the Poly Item ItemType (Part, Document or CAD Document).

Try it … Select a Poly Item

1. Create a new Problem Report (PR) from the Change Management category on the TOC.
2. Click the Find Item button in the Affected Item field to display the Select Items window.
3. Open the dropdown menu for the Type field and select one of the Poly Sources.
4. Click Cancel and Delete the new PR item.



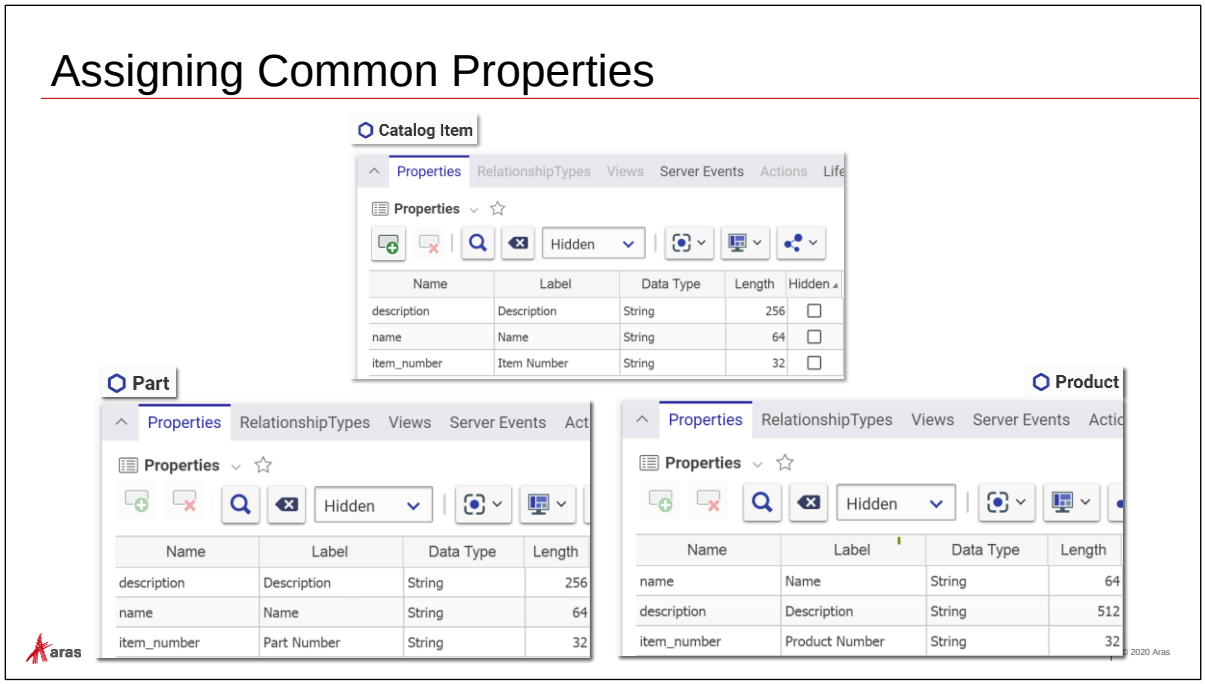
Creating a New Poly Item ItemType

Define Poly Items by creating an ItemType using the Poly Item Implementation Type. You can then define which Poly Sources represent the Poly Item. In this example, you create a new Poly Item named Catalog Item that represents both the Part and Product ItemTypes.

When you create a Poly Item ItemType, a List Item is created automatically with the same name as the Poly Item ItemType. The List is used for the definition of the List Property Type. The List can be viewed from the Administration Category of the TOC.

Try it … Create a Poly Item ItemType

1. Navigate to TOC > Administration > ItemTypes and create a new ItemType.
2. Name the ItemType *Catalog Item* and provide the appropriate values as shown above.
3. Select Poly Item as the implementation type, clicking OK to acknowledge the warning about this step being irreversible.
4. Save the ItemType, open the Poly Sources relationship tab, and select the existing **Product** and **Part** ItemTypes.
5. Click Save.



Assigning Common Properties

Now you define which shared Properties can be viewed in search grids. The Poly Item ItemType can be assigned additional properties that are common to the Poly Source ItemTypes. Note that the Property Name spelling must be the same for all Poly Source and Poly Item ItemTypes. The Property Data Types must also be compatible.

Try it … Assign Common Properties

In this exercise, the item_number, name, and description properties are defined in the Poly Item ItemType.

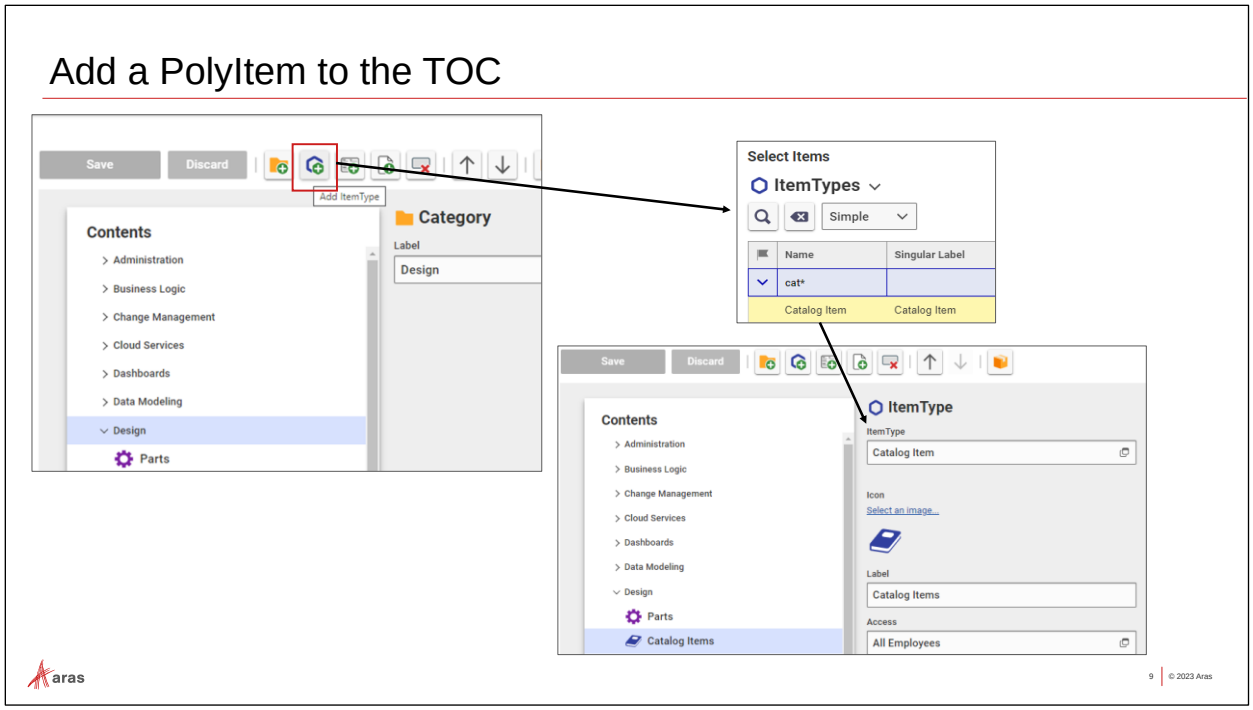
1. Open for editing the new **Catalog Item** ItemType.
2. Create three new properties: item_number, name, and description.

Name	Label	Data Type	Length
item_number	Item Number	String	32
name	Name	String	64
description	Description	String	256

3. Make sure that the properties are displayable by clearing the Hidden checkbox.
4. Click Done.

Note

The spelling of the name of each property must be the same for all related ItemTypes and the Data Type of their properties must be compatible. Their labels can be different.



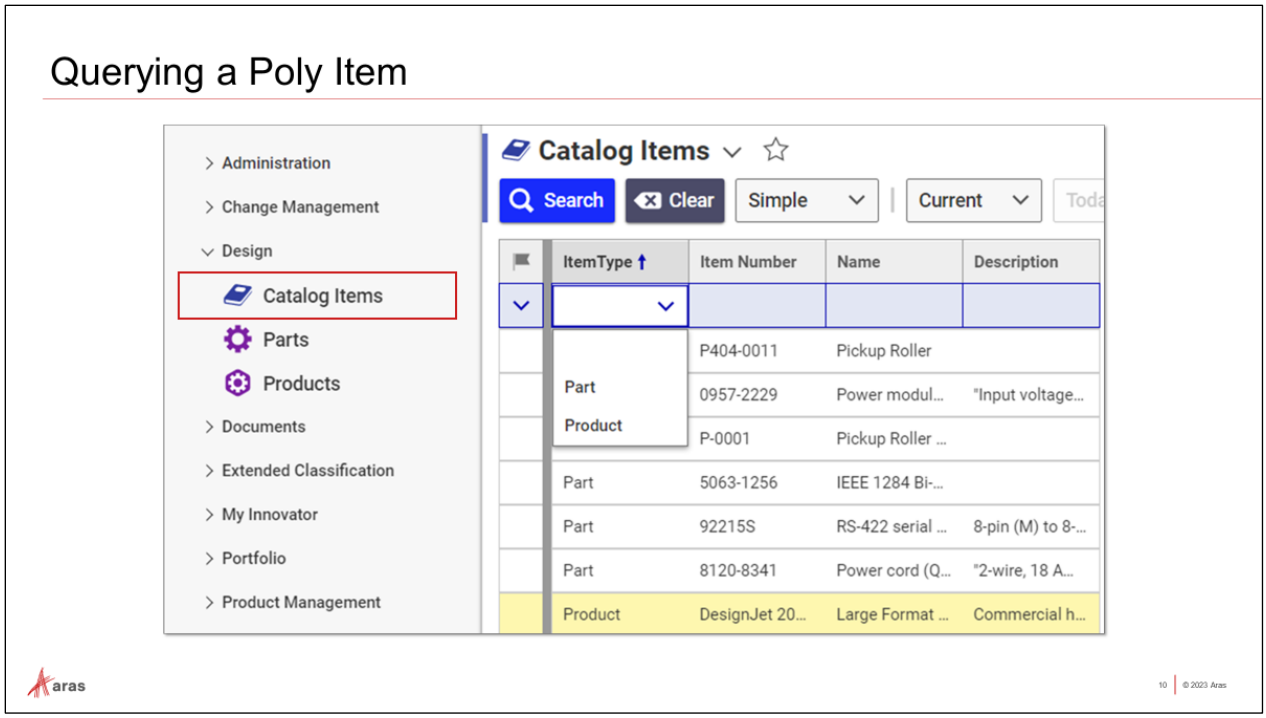
Add a Poly Item to the TOC

Administrators can use the TOC Editor to add Poly Items to the TOC. Use the Add ItemType button to add Poly ItemTypes in the same way you can add Single ItemTypes.

Try it … Add the Catalog Items Poly Item to the TOC

Using your knowledge of the TOC Editor, add the **Catalog Items** (Poly Item) to the Design category of the TOC with **All Employees** access.

1. Open the TOC Editor.
2. Select the Design category and click the Add ItemType button.
3. Search for and select the *Catalog Item* ItemType and click OK.
4. Accept the default Label value and change the Access to *All Employees*.
5. Click Save.

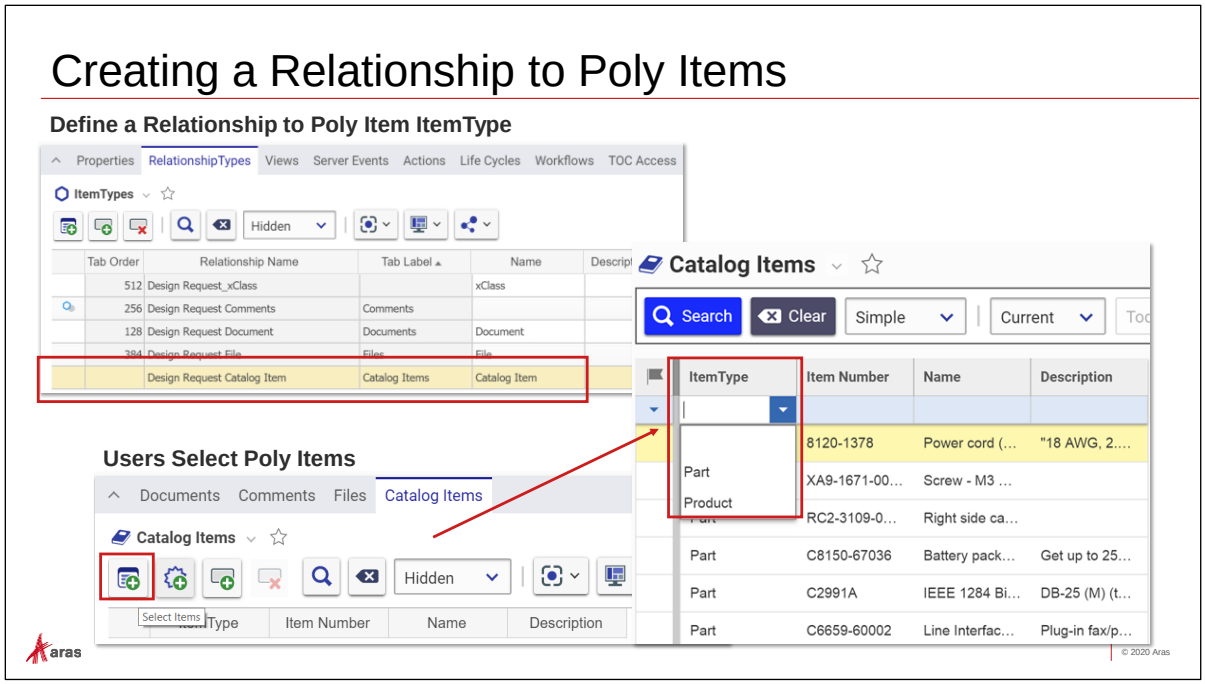


Querying a Poly Item

When you search for Poly Items, a search grid displays each unhidden property from the Poly Sources. The list that is automatically created with the Poly Item is also displayed in the search grid, allowing the user to filter by the ItemTypes in the Poly Sources tab.

Try it … Query a Poly Item

1. As Terry Adams, go to the TOC > Design > Products, and create a new Product.
2. As your student login, go to the TOC > Design > Catalog Items and open a search grid.
3. Use a Simple Search to locate either Parts, Products, or both.



Creating a Relationship to Poly Items

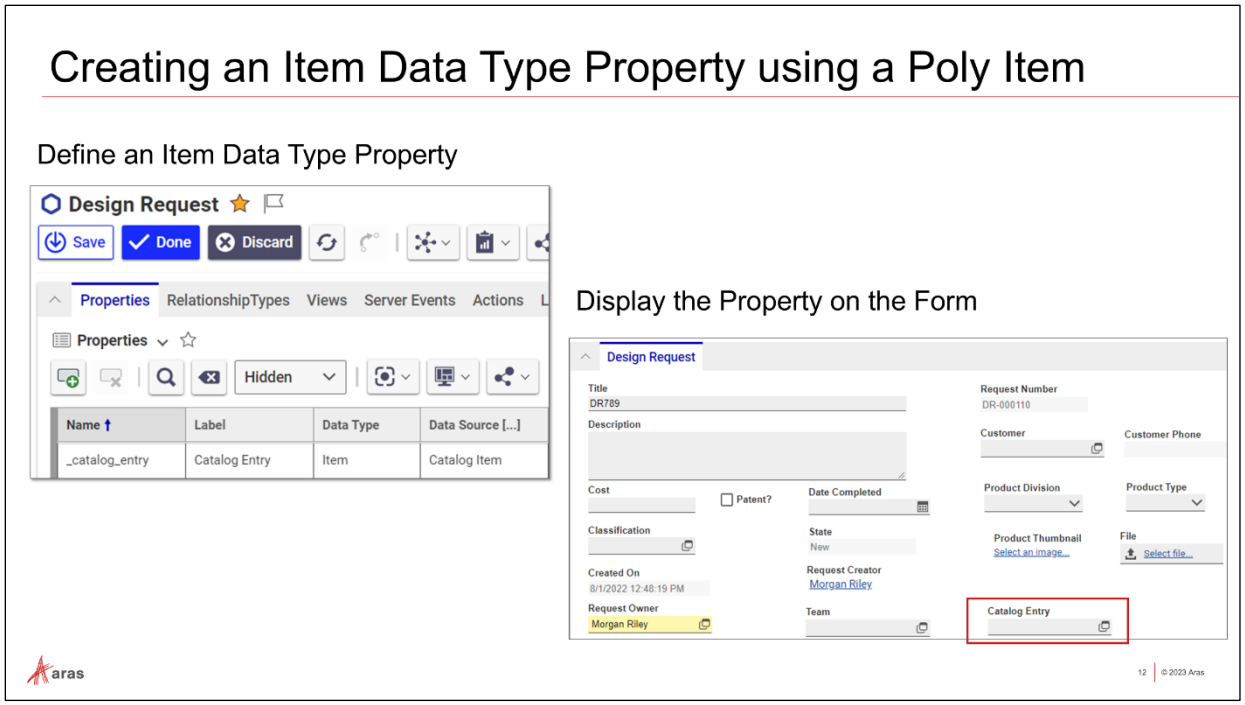
You can establish a relationship to a Poly Item ItemType to allow users to select from a collection of ItemTypes in a relationship grid. In the example above, a new relationship is created on the Design Request ItemType to allow users to select a Poly Item as a related item.

Try it … Create a Relationship to a Poly Item

1. In TOC > Administration > ItemTypes, open the **Design Request** ItemType for editing and select the RelationshipTypes tab.
2. Create a relationship to the new itemType (Add ItemTypes button) and select the **Catalog Item** as the related item. Be sure to add a Tab Label.

Relationship Name	Tab Label	Name
Design Request Catalog Item	Catalog Item	Catalog Item

3. Click Done.
4. Open for editing the new RelationshipType and select both *Requires Related* and *Open Related Form*.
5. Click Done.
6. Test the new Relationship by editing a Design Request. Select the Catalog Items Relationship tab and click the Add Catalog Items button to choose Parts and/or Products.



Creating an Item Data Type Property using a Poly Item

A property of Data Type Item can use a Poly Item ItemType. In the example above, a new property named _catalog_entry is defined in the Design Request ItemType and added to the Form. Users are then able to select from a collection of Items of all Poly Sources configured on the Poly Item ItemType.

Try it … Create an Item Data Type Property using a Poly Item

1. On the **Design Request** ItemType, create a new property:

Name	Label	Data Type	Data Source […]
_catalog_entry	Catalog Entry	Item	Catalog Item

2. Click Done.
3. Modify the **Design Request** Form to add the property to display on the form.
4. Also add it to the other forms related to the **Design Request** ItemType.
5. Test the Poly Item Property (CatalogItem) by editing a Design Request and using the Catalog Entry property to choose a Part or a Product.
6. Click Done.

Summary

In this unit, you learned how to define and create Poly Items. You should now be able to:

- Define the purpose of a Poly Item
- Use Poly Items for establishing Relationships or Item Data Type Properties
- Create a Poly Item as a Property

Review Questions

1. How do you configure which ItemTypes will define a Poly Item?

2. What are the three use cases for a Poly Item?

3. What ItemType features are not available on a Poly Item?

4. Why would you configure a relationship related item as a Poly Item?

Lab Exercise

Goal

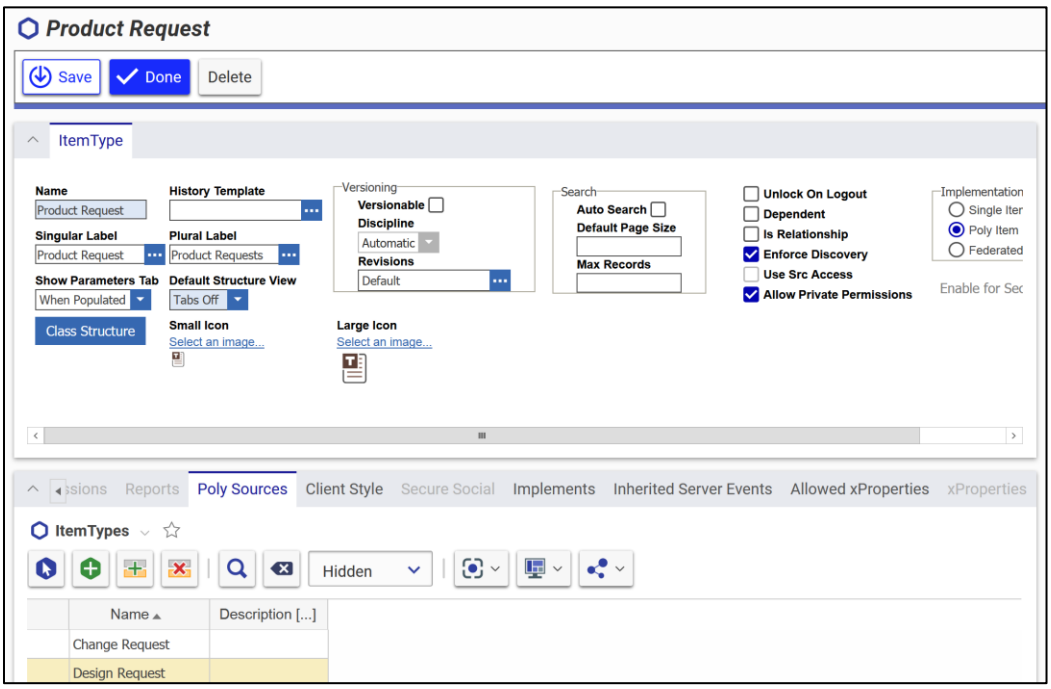
Create a new Poly Item ItemType using the ItemTypes you created in this course.

Scenario

In this exercise, you create a new Poly Item ItemType named Product Request that allows a user to query both the Design Requests and the Change Requests you configured earlier in the course in a single search grid.

Steps

1. Create a new ItemType named **Product Request** and provide the appropriate labels and icons.
2. Choose the implementation type of Poly Item.
3. Select the Design Request and Change Request ItemTypes as Poly Sources for the new ItemType.



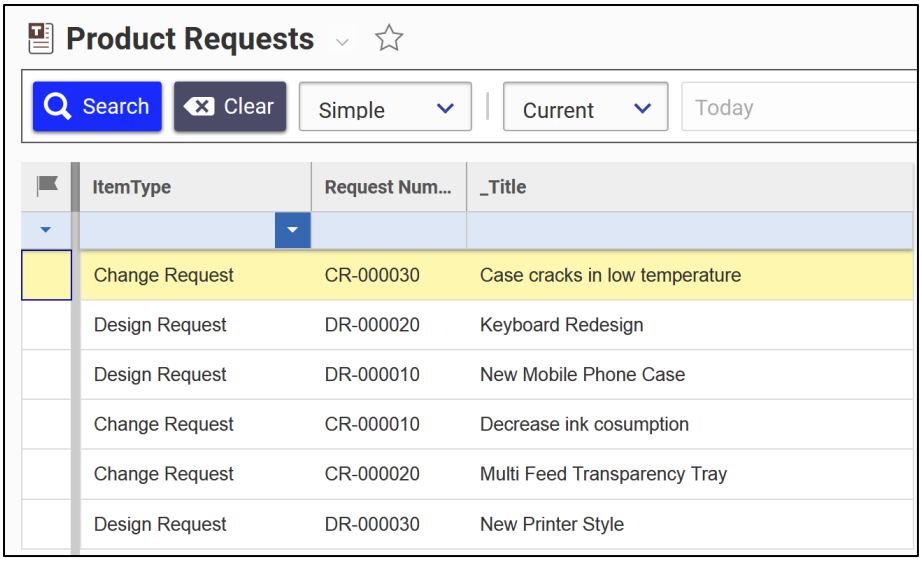
4. Save changes to establish the Poly Item ItemType.
5. Locate the Properties tab and define the following properties that are common to both Poly Source ItemTypes and then save the changes.

   Note: If you did not build the Properties for the Design Request ItemType with the instructor earlier, please do it now.

Name	Label	Data Type	Length
_item_number	Request Number	String	32
_title	Title	String	64

6. Click Done.
7. In the TOC Editor, add the **Product Request** ItemType in the *Product Management* category, with access to World.
8. Locate the new ItemType in the TOC and run a query to display the results.

   The following example is sorted by Title and your results may differ.


Product Requests
Search
Clear
Simple
Current
Today
ItemType
Request Num...
_Title
Change Request
CR-000030
Case cracks in low temperature
Design Request
DR-000020
Keyboard Redesign
Design Request
DR-000010
New Mobile Phone Case
Change Request
CR-000010
Decrease ink cosumption
Change Request
CR-000020
Multi Feed Transparency Tray
Design Request
DR-000030
New Printer Style

Unit 20 Configuring Command Bars

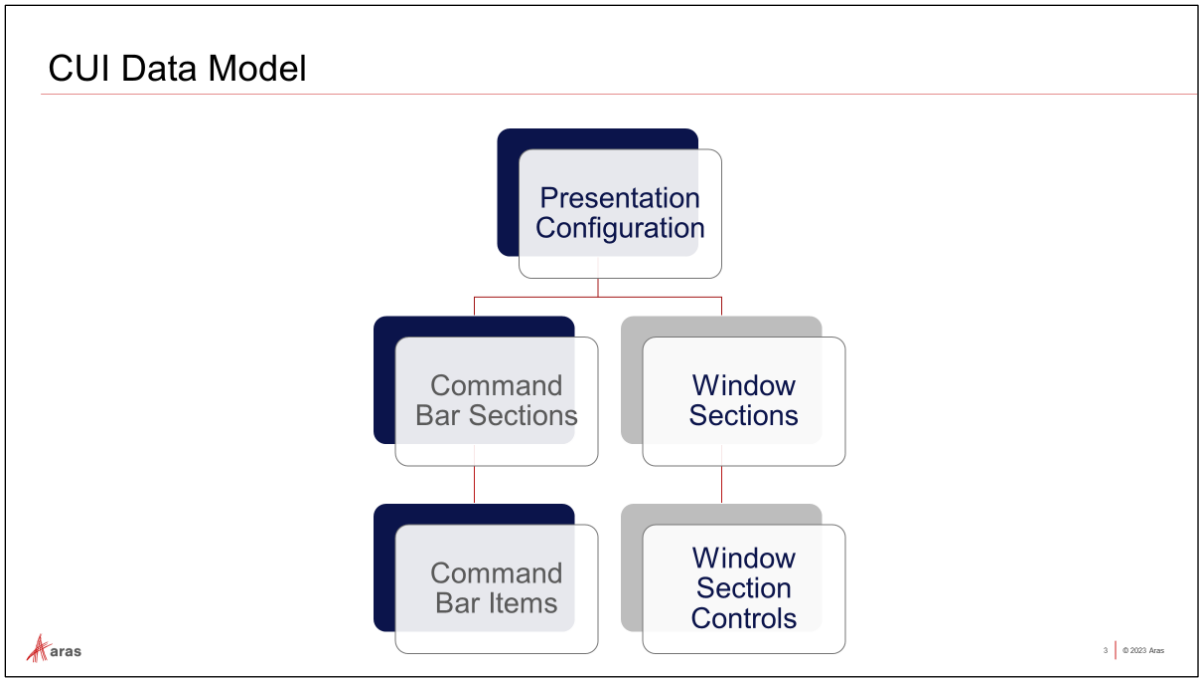
Overview

The Aras Innovator client uses configurable command bars that allow you to add, remove and revise buttons, menus, and associated windows shortcut keys without any programming.

In this unit, you learn how to add new command buttons and shortcut keys that are defined in the Configurable User Interface (CUI) data model.

Objectives

- Introduce the Configurable User Interface (CUI) Data Model
- Review Command Bar Section Locations
- Identify Command Bar Item Actions
- Understand Command Bar Item Action Ranking and Security
- Add a Command Bar Button and Windows Shortcut



CUI Data Model

The Configurable User Interface (CUI) allows an Aras Administrator to define the layout and behavior of the Aras Innovator client application. The configuration is based on a hierarchy of ItemTypes that define both the windows design as well as the individual controls that allow a user to interact with the application.

Presentation Configuration

A global Presentation Configuration acts as a container for the CUI configuration. The global presentation applies to all items in the application, but it can be overwritten or augmented by an ItemType presentation which is specific to a category of items (e.g., Part). One global presentation defines the standard out-of-the-box Aras web client. The ItemType presentation is outside the scope of this course.

Command Bar Sections

Command Bar Sections define the content of a window control and contain a collection of one or more command bar items.

Command Bar Items

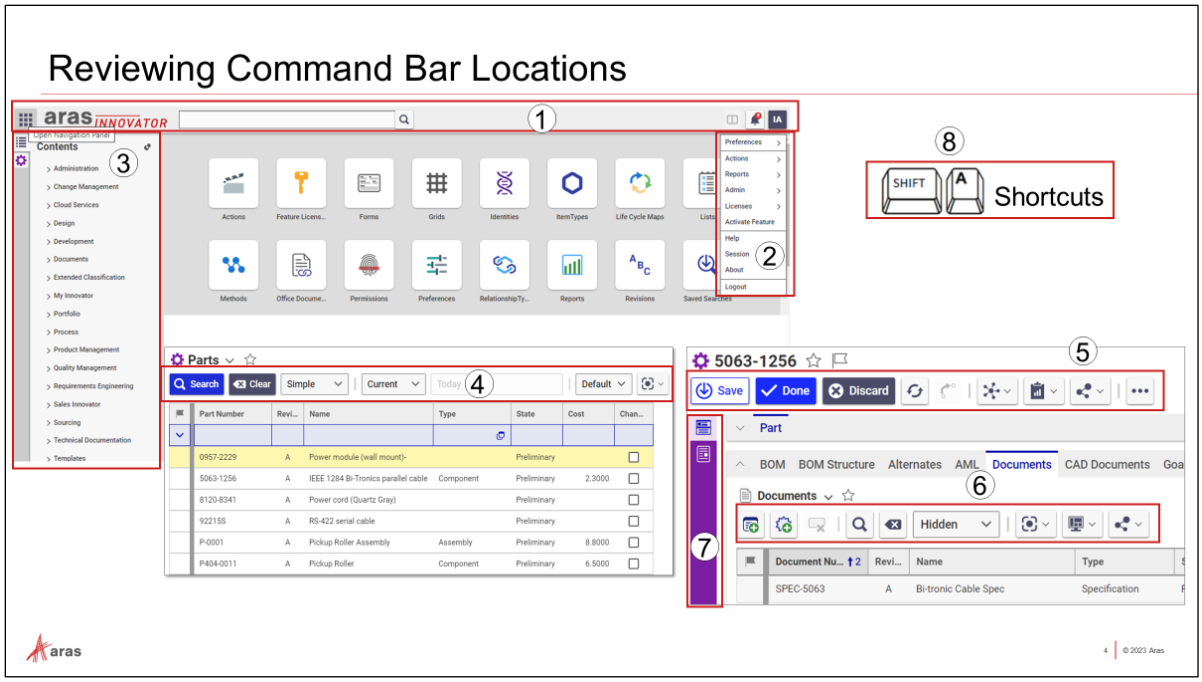
Command Bar Items define each individual control contained in a command bar section. Examples include toolbar menus and buttons, windows shortcut keys, and context menu entries.

Window Sections

A Window Section defines the overall layout of an item window and contains various window controls that display on the screen. Modifying window sections is outside the scope of this course.

Window Section Controls

Modifying window section controls is outside the scope of this course. They define the layout of a title bar, toolbar, tab control, form accordion element, and a relationship accordion element.



Reviewing Command Bar Locations

Configurable command bar sections are defined for many locations in the Aras Innovator client. This unit focuses on the primary locations.

① **Main Window Header**
Contains the top navigation panel.

② **Main Window Header User Menu**
Displays menu options available from the User Menu.

③ **Table of Contents (TOC)**
Defines the configured categories and labels.

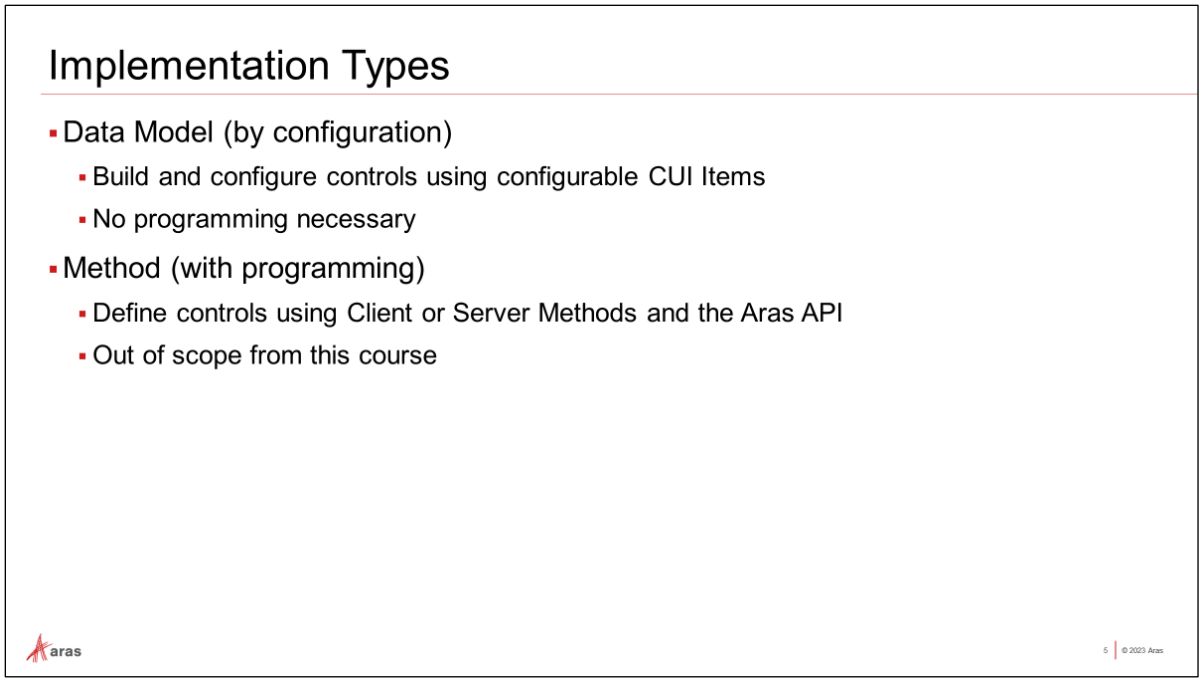
④ **Search View Command Bar**
Displays buttons and controls used in the main search grid.

⑤ **Item View Command Bar**
Defines the buttons and menus displayed on the item window.

⑥ **Item View Relationships Command Bar**
Defines the controls used in the item relationship (tabbed) grid.

⑦ **Item View Sidebar**
Displays the buttons used in the sidebar.

⑧ **Item Window Shortcuts and Main Window Shortcuts**
Defines windows shortcut keys that can be associated with a Method to perform an operation.



Implementation Types

Customization of the user interface is accomplished in two ways: either through the Data Model or through Methods.

Data Model

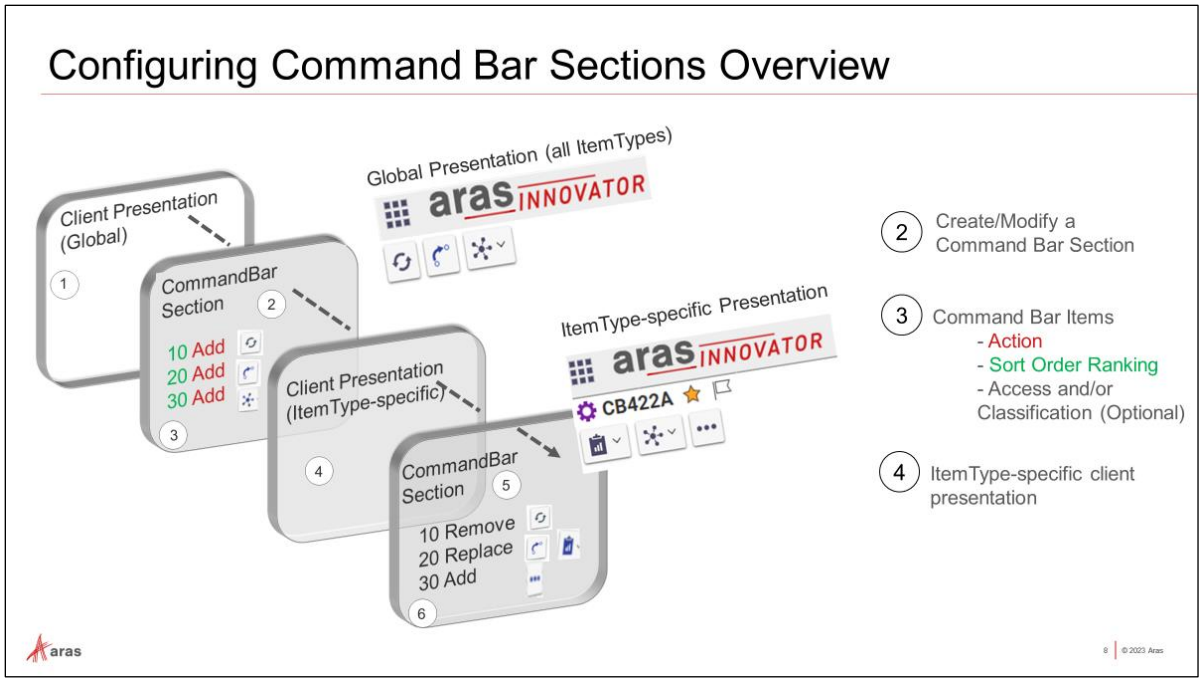
A reserved set of ItemTypes provides the ability to add, revise and remove menu entries and command bar buttons in the locations described on the previous page. Method code is not required to create new controls using the data model implementation. Controls can also be configured to determine who should have access to the controls and when they should appear.

In this course, we will make changes to the existing data model to create new controls.

Methods

The Aras API also allows UI controls to be created dynamically with programming logic using either a client or server Method. This approach is best when the access to a control is based on complex business rule(s).

Using Methods to create UI controls is outside of the scope of this course.



Configuring Command Bar Sections Overview

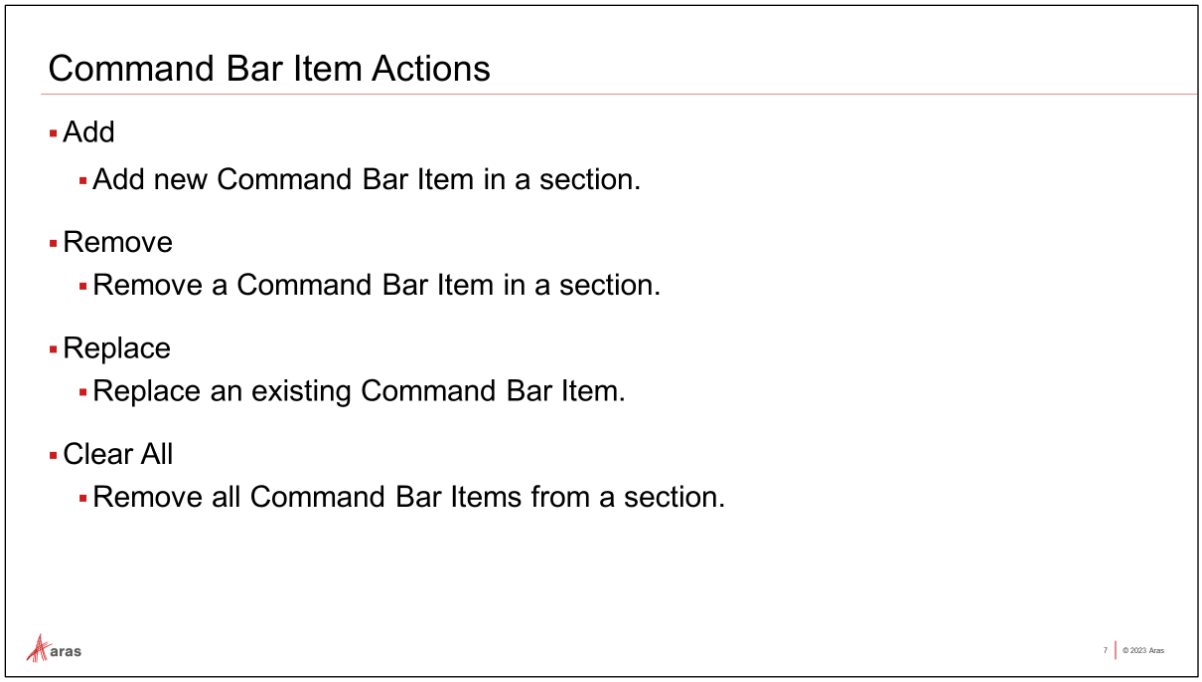
Command bars are configured by defining (or modifying) a command bar section that contains Command Bar Items (e.g., menus, buttons).In this example, the command bar section is the Item View Command Bar.

You add, remove, or revise a command bar item using an Item Action.

The appearance and order of the command bar items are determined by a sort order ranking.

Command bar items must be qualified by identity. They can be linked to an item classification so that a command bar item only appears for a certain group of users, based on the current classification of the item.

The global client presentation contains the command bars applying by default to all ItemTypes. Additionally, specific Command Bars can be created for specific ItemTypes.



Command Bar Item Actions

Four actions are available when configuring a control.

Add

Adds a new UI control to a Command Bar Section.

Remove

Removes a UI control that was previously added in another Command Bar Section.

Replace

Replaces a UI control that was previously added in another Command Bar Section.

Clear All

Removes all controls from a Command Bar Section.

Action Ranking

- Each CUI Item Action is assigned a sort order value
- Actions are ranked and then executed in sort order
- Sort order allows for positioning of controls in a section
- Sort order allows for “override” of previous configurations based on rank order

ItemType	Name	Sort Order	Action
Button	itemview.itemcommandbar.default.save	128	Add
Button	itemview.itemcommandbar.default.done	256	Add
Button	itemview.itemcommandbar.default.delete	384	Add
Button	itemview.itemcommandbar.default.edit	416	Add

aras

8 | © 2023 Aras

Action Ranking

Actions are ranked by a sort order that indicates the order of execution within a section. The lowest order number executes first and so on.

In the case of the Add action, this dictates the position order of a command bar item or window section.

In the example above, a series of buttons are added to a toolbar in the sequence indicated by the sort order number.

Qualifying Identity Access and Classification

- CUI Item Actions can be qualified by Identity
  - The Command Bar Item Action will only execute if the currently logged in user belongs to the declared Identity
- Actions can also be qualified to run based on the value of an associated ItemType classification

Sort Order	Action	For Identity [...]	For Classification
32	Add	World	
48	Add	World	
64	Add	World	
80	Remove	Engineering	Product
96	Add	Manufacturing	Component
112	Add	CRB	

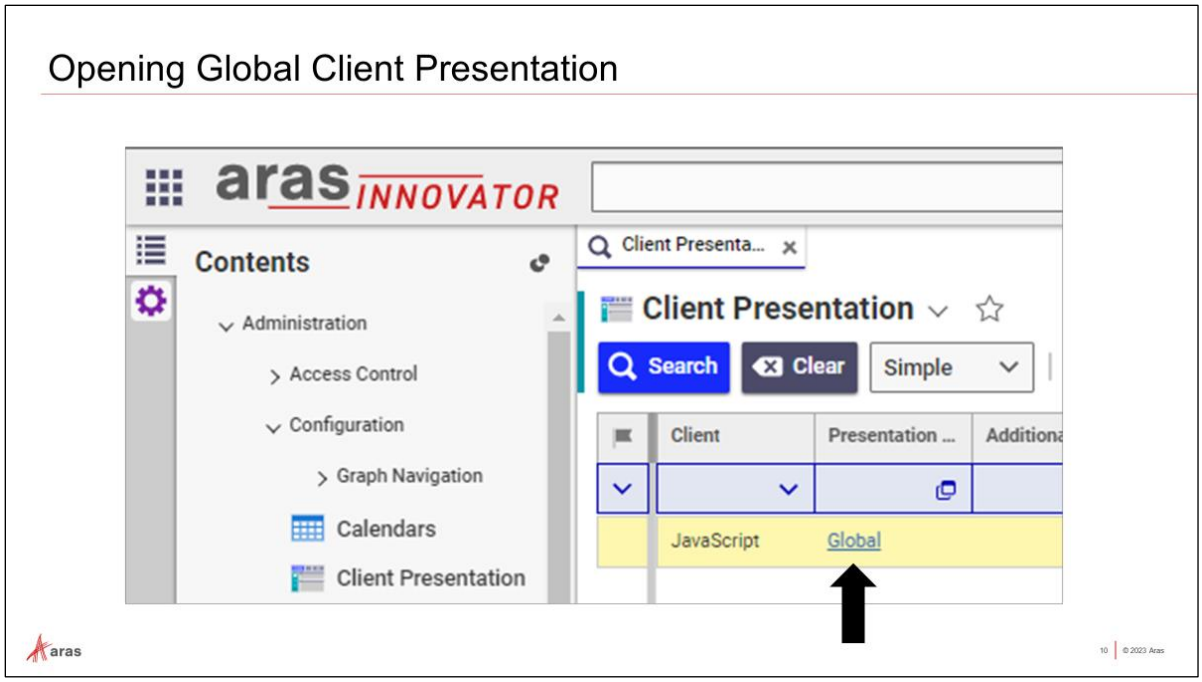
aras

9 | © 2023 Aras

Qualifying Identity Access and Classification

In addition to order ranking, each action can be qualified by an identity and an item classification. At runtime, the qualifiers are evaluated to determine whether the action should be executed for the current user on the current item.

In the example above, the first three Add actions will execute regardless of the user logged in to Aras Innovator (World). The Remove action will only execute if the user is a member of Engineering and the item is classified as Product. The next Add action will only occur for members of Manufacturing on an item classified as Component. The final Add action occurs if user is a member of the CRB identity.

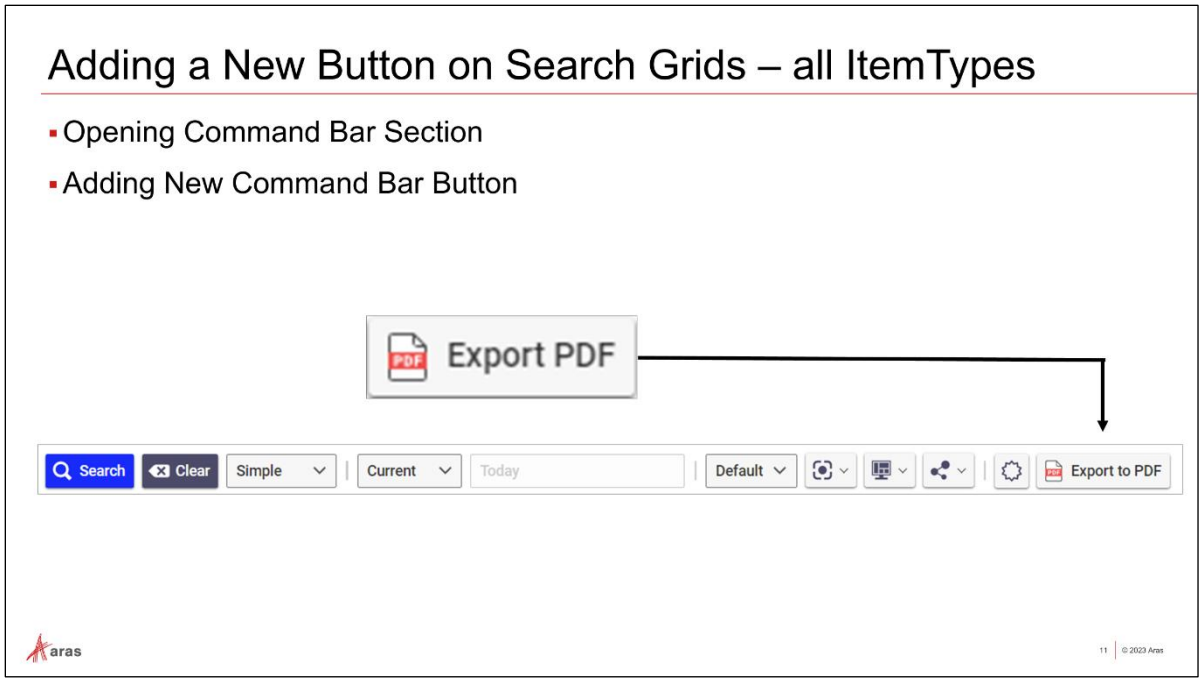


Opening Global Client Presentation

The global client presentation defines the default user interface for all items in the Aras Innovator application. There is only one global presentation for the standard Aras Innovator client, but additional presentations are possible for custom client implementations.

Try It … Open the Global Presentation

1. Login as an Administrator (your student login).
2. Navigate to Administration > Configuration > Client Presentation and run a full search.
3. Click the **Global** link in the Presentation column to open the presentation item.

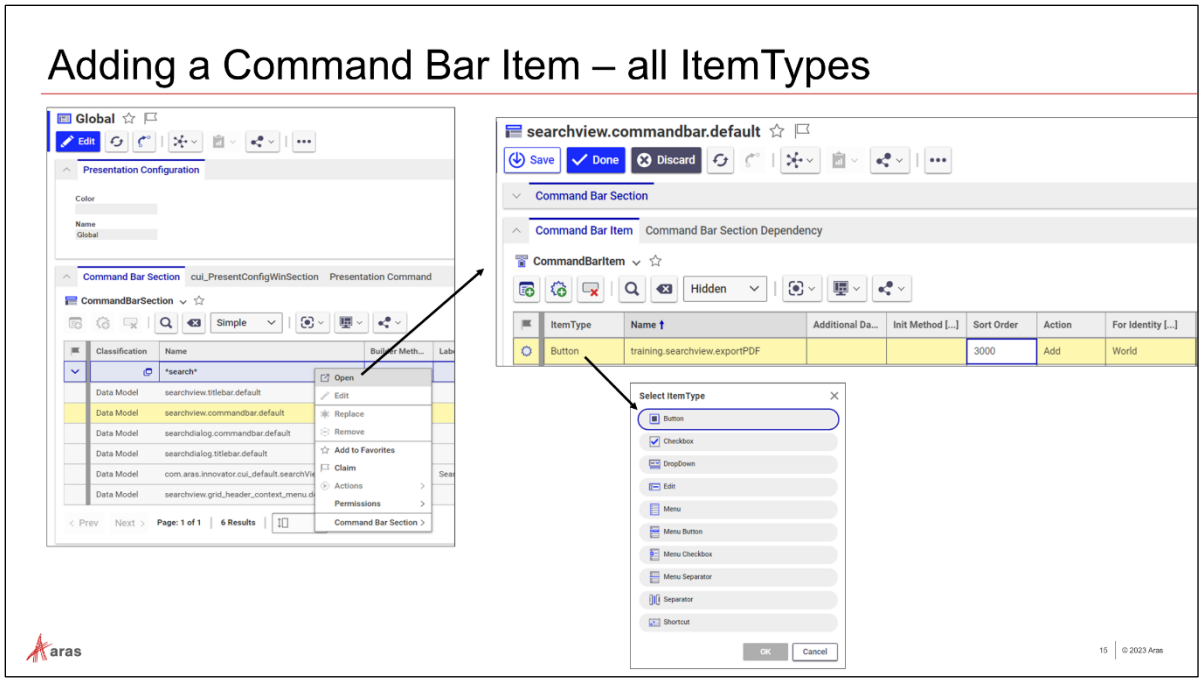


Adding a New Button on Search Grids – all ItemTypes

Toolbar buttons are represented as command bar items in the CUI. Each button is positioned using sort order ranking. In this example, a new button will be defined for the search grid of all ItemTypes to allow a user to export the contents of search grids to a PDF file.

To Add a New Button

- Locate and open the desired command bar section.
- Create a new command bar item.



Adding a Command Bar Item – all ItemTypes

The Command Bar Section representing the search grid toolbar is named **searchview.commandbar.default**.

Try It … Open a Command Bar Section

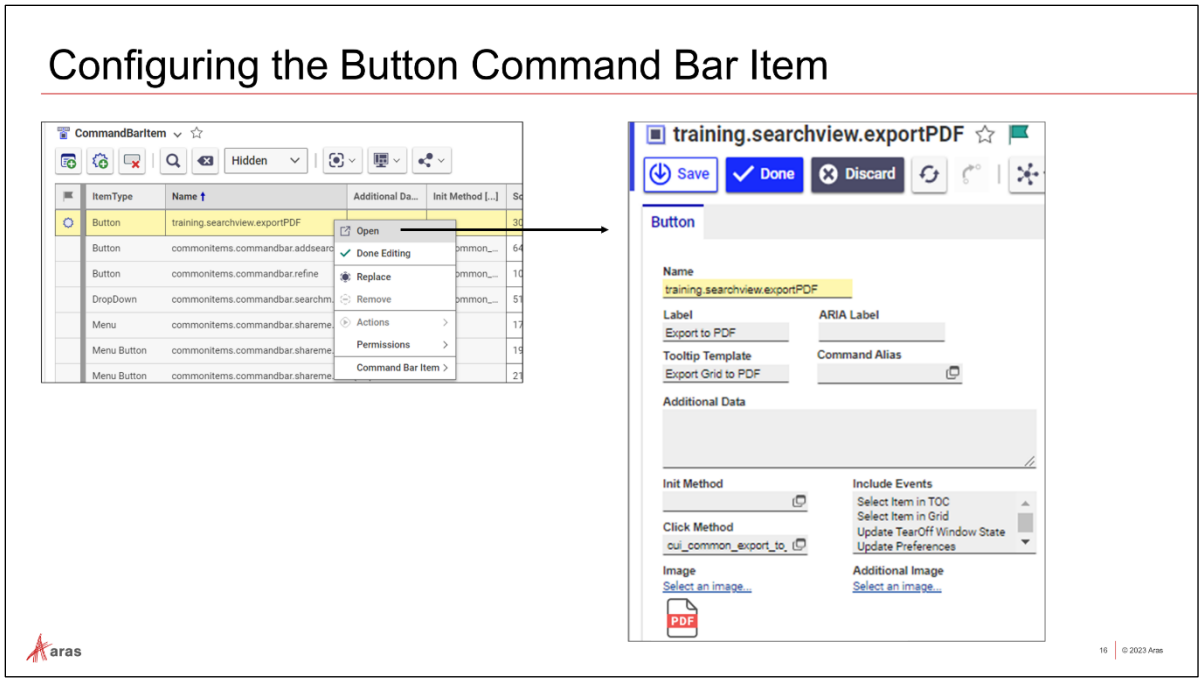
1. In the presentation configuration item, locate the Command Bar Section tab.
2. Set the search mode to Simple.
3. Click the Clear Search Criteria button, enter **search** in the Name column and click Search.
4. Widen the Name column, right-click and open **searchview.commandbar.default** from the context menu.

Try It … Add a Button Command Bar Item

1. With the **searchview.commandbar.default** tab open, click the Edit button.
2. Click the New Command Bar Item button in the relationship grid.
3. Choose **Button** from the Select Item Type dialog window and click OK.
4. Enter the following information into the new row:

Name	training.searchview.exportPDF
Sort Order	3000
Action	Add
For Identity…	World

   The new Button Command Bar Item will appear at the bottom of the list.
5. Right-click on the row and select Open to continue configuring the new command button.



Configuring the Button Command Bar Item

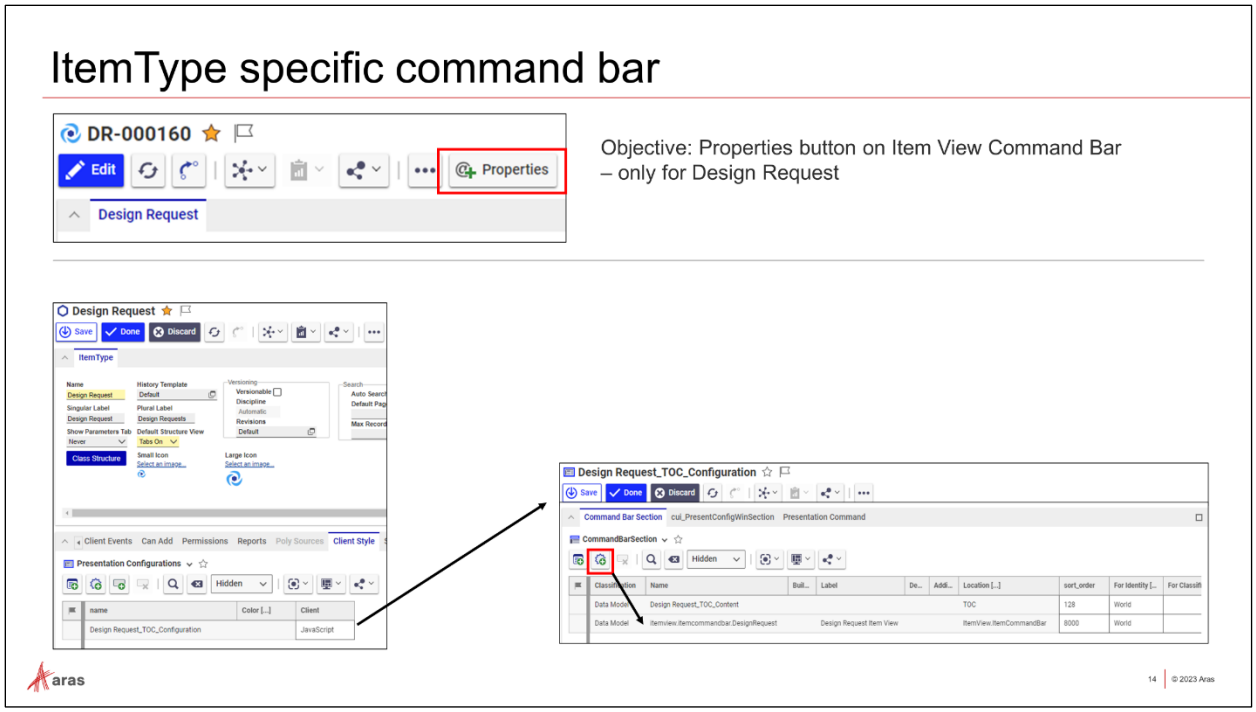
The command bar item defines through the Click Method, what Method will execute when the button is clicked by a user, as well as the label and icon to be displayed on the button control. The Tooltip Template displays when a user hovers over the button.

Try It … Configure the Button Command Bar Item

1. Enter in following:

Label	Export to PDF
Tooltip Template field	Export Grid to PDF
Click Method	cui_common_export_to_pdf_click

2. Select an image to display (optional) on the button.
3. Click Done to save the Button configuration.
4. Go back to the previous tab (the Command Bar Section) and Click Done to capture the changes.



ItemType-specific command bar

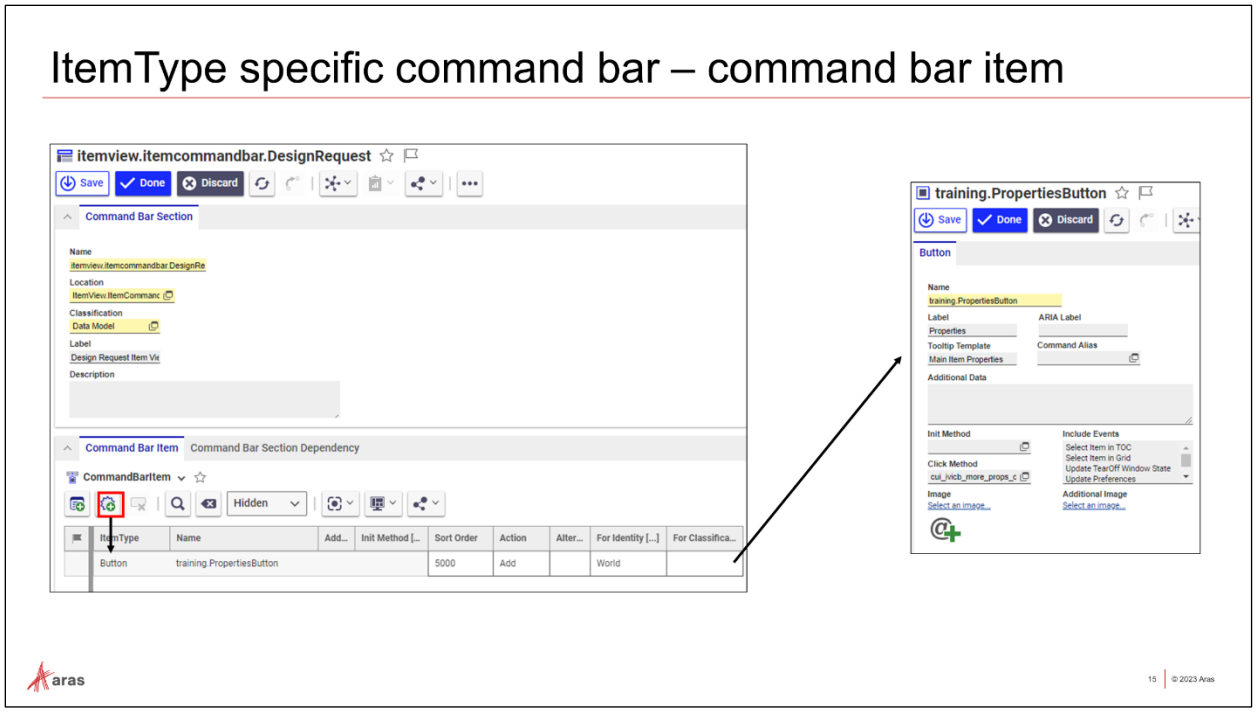
In addition to the Global Client Presentation, which applies to all ItemTypes by default, an administrator can also configure a UI specific to an ItemType (command bar section or window section). This specific UI will complement the default UI as an extra layer. This is configured in the ItemType relationship tab *Client Style*. The Location property is used by the system to add up the different settings on different command bar sections.

If no presentation configuration exists, create one using Client=JavaScript. Only one presentation with that Client value should exist in an ItemType relationship. A presentation configuration can already exist in Client Style, for example in case the ItemType was added in the TOC.

Try It … Configure an ItemType-specific Command Bar section

1. Open the **Design Request** ItemType for viewing.
2. Go to the Client Style relationship tab, and right-click / Open on the Design Request_TOC_Configuration Presentation Configuration. (Note: in case no Presentation Configuration exists, you need to create one by clicking “New Presentation Configuration” or adding the Design Request ItemType in the TOC through the TOC Editor).
3. Edit the Design Request_TOC_Configuration Presentation Configuration, and in the Command Bar Section tab, click “New Command Bar Section”. Enter in following:

Classification	Data Model
Name	itemview.itemcommandbar.DesignRequest
Label	Design Request Item View
Location	ItemView.ItemCommandBar
Sort Order	8000
For Identity…	World



ItemType-specific command bar – command bar item

Try It … Configure the Button Command Bar Item

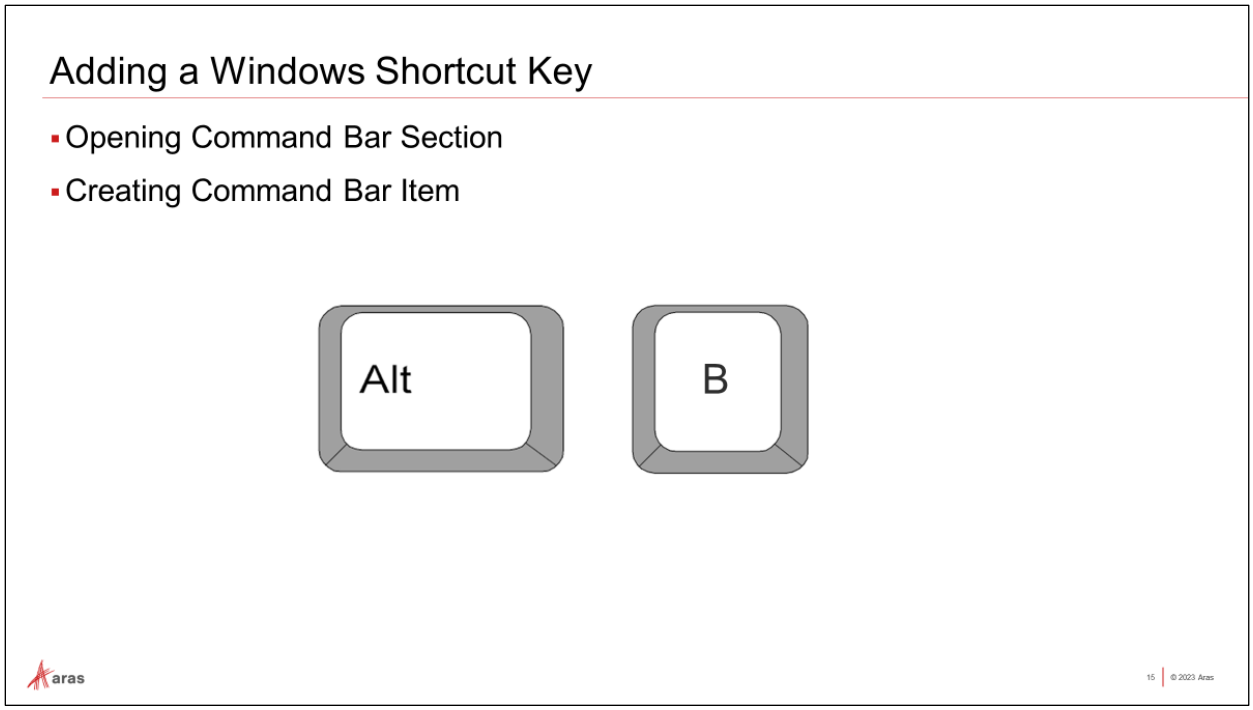
1. Right-click on **itemview.itemcommandbar.DesignRequest** Command Bar Section, then Open.
2. Click Edit.
3. In *Command Bar Item* tab, click **New Command Bar Item**.
4. In the *Select ItemType* window, choose **Button**, then click OK.
5. Enter the following:

Name	training.PropertiesButton
Sort Order	5000
Action	Add
For Identity	World

6. Right-click on the *Button* command bar item, then Open.
7. Enter the following:

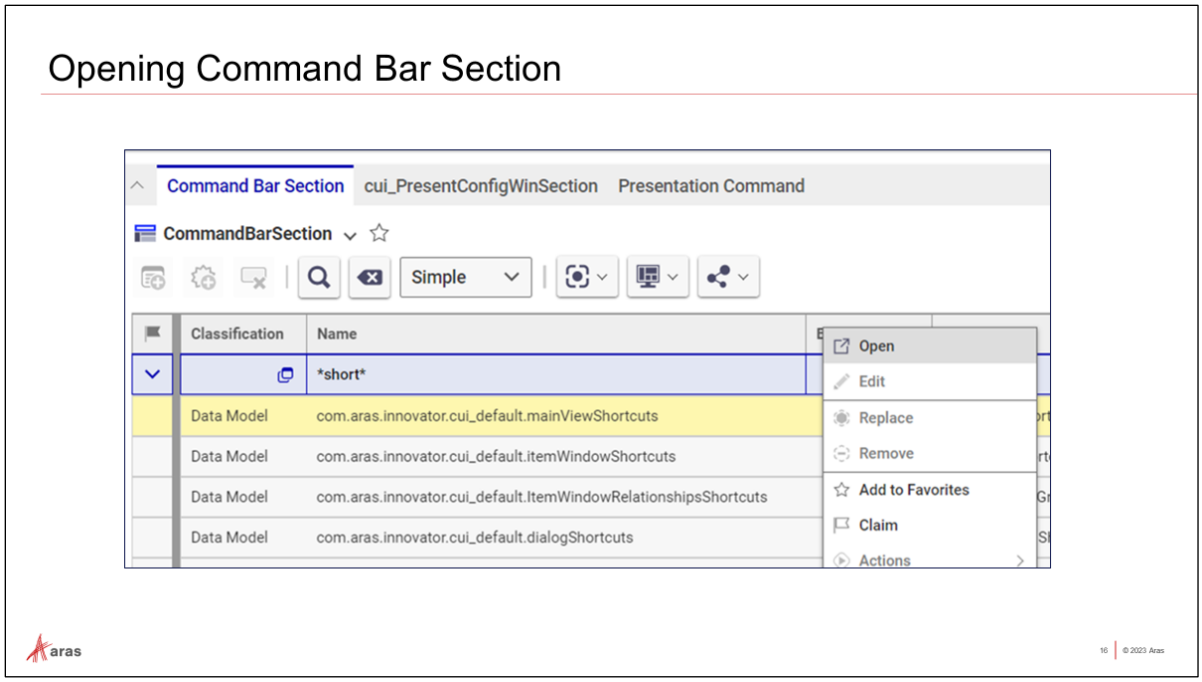
Label	Properties
Tooltip Template	Main Item Properties
Init Method	(empty)
Click Method	cui_ivicb_more_props_click
Image	(select an icon of your choice)

8. Click Done on the Button command bar item and close that Aras tab.
9. Click Done on the command bar section and close that Aras tab.
10. Click Done on the Design Request_TOC_Configuration Presentation Configuration.



Adding a Windows Shortcut Key

Shortcut keys allow a user to access a feature in Aras Innovator without the use of a mouse. You can individually create shortcuts for different locations.

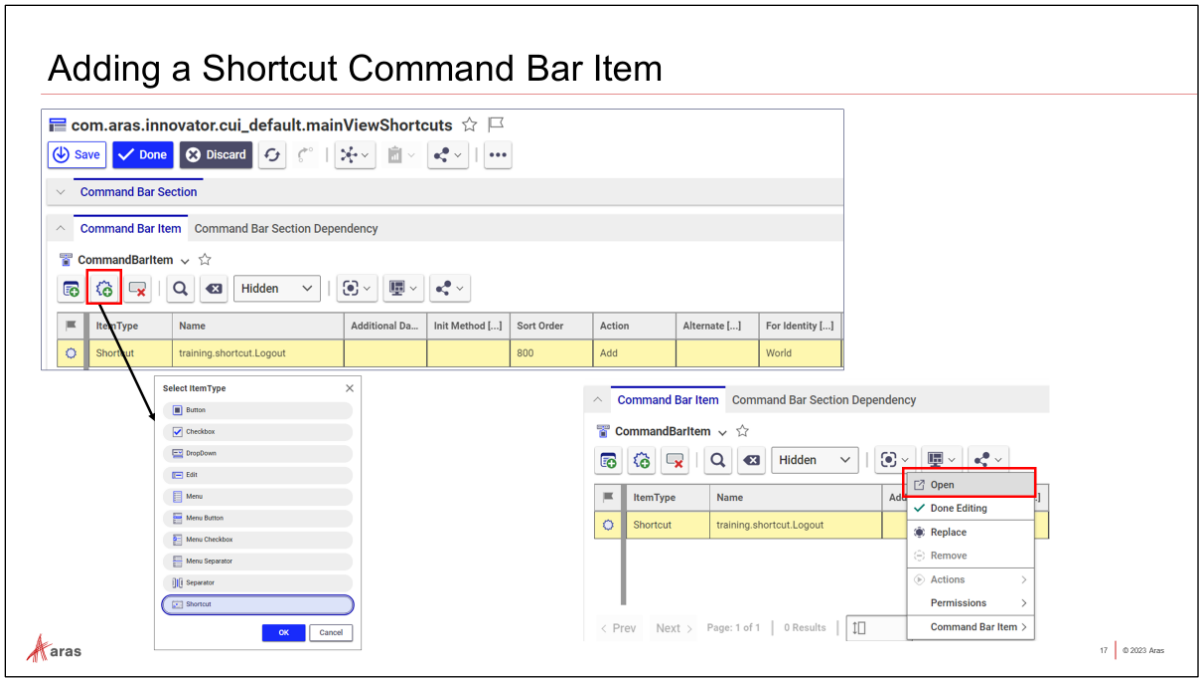


Opening Command Bar Section

Shortcut keys are contained in a command bar section that represents a location in the Aras Innovator client. Aras Innovator users would like to quickly logout so we will add a logout shortcut.

Try It … Open Command Bar Section

1. Open the Global Client Presentation Command Bar Section relationship tab and ensure it is set to the Simple Search mode.
2. Clear the current Search criteria, enter **short** in the Name search filter field, and run the search.
3. Widen the name field, locate the command bar section **com.aras.innovator.cui_default.mainViewShortcuts,** right click, and choose Open to view the section.



Adding Shortcut Command Bar Item

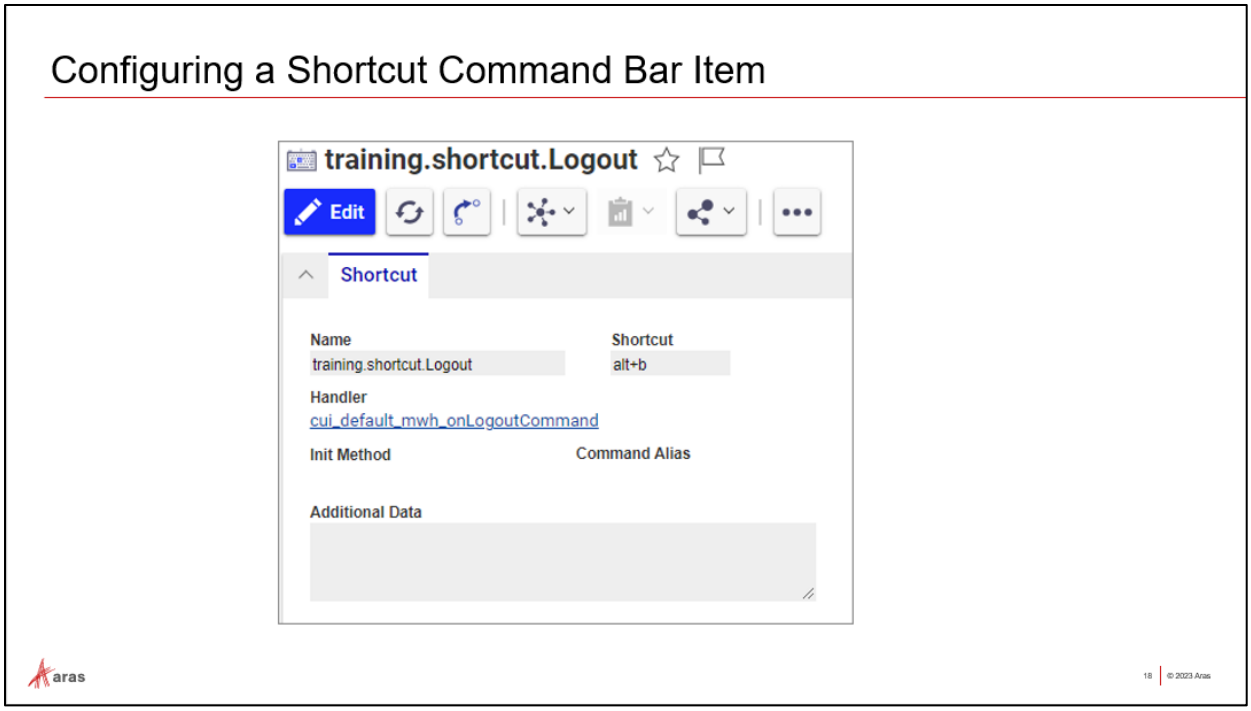
Shortcut keys are represented as command bar items in the database with the ItemType of Shortcut.

Try It … Add a Shortcut Command Bar Item

1. With **com.aras.innovator.cui_default.mainViewShortcuts** open in a new tab, click the Edit button.
2. On the Command Bar Item Relationship tab, click the New Command Bar Item button.
3. Choose **Shortcut** from the Select ItemType dialog window and click OK.
4. Enter the following information into the new row:

Name	training.shortcut.Logout
Sort Order	800
Action	Add
For Identity…	World

5. Right click on the row and select Open to continue configuring the new shortcut.



Configuring the Shortcut Command Bar Item

The shortcut command bar item is configured with one or more keyboard keys (supported combinations are shown below). The Handler determines what program that will execute when the key(s) are pressed.

For any combination of shortcut keyboard keys, they must be in the following order:
Alt then **Ctrl** then **Shift**.

Supported Keyboard Combinations

alt+	ctrl+	shift+	ctrl+shift+	Button on keyboard
Supported	Supported	Supported	Supported	a-z
Supported	Supported	Supported	Supported	0-9
Supported	Supported	Supported	Supported	Num0-Num9
Supported	Supported	Supported	Supported	~
Supported	Supported	Supported	Supported	tab
Supported	Supported	Supported	Supported	enter
Supported	Supported	Supported	Supported	insert
Supported	Supported	Supported	Supported	delete
N/A	Supported	N/A	N/A	pageup, pagedown, home, end
N/A	Supported	N/A	N/A	F1-F12
N/A	Supported	N/A	N/A	arrows, Num-arrows
N/A	Supported	N/A	N/A	<, >
N/A	Supported	N/A	N/A	[,]
N/A	Supported	N/A	N/A	-, +, Num-, Num+

Try It … Configure a Shortcut Command Bar Item

Warning: in this try-it, make sure to fill the **Handler** field and not the **Init Method** field, otherwise your environment would logout at any action.

1. With the **training.shortcut.Logout** open for editing in a new tab, enter the following:

Shortcut	alt+b
Handler	cui_default_mwh_onLogoutCommand
Init Method	(empty)

2. Click Done. Close that tab.
3. Return to the **com.aras.innovator.cui_default.mainViewShortcuts** Command bar Section tab and click Done. Close that tab.
4. To test your new shortcut, log out and then log back in again.
5. Once you login, use *alt+b* to logout of Aras Innovator.
6. After logging in again, test the Export to PDF button, by opening a search grid on any ItemType (e.g. Parts) and click your new button. The PDF is exported according to your browser settings.
7. Similarly, open an existing Design Request Item, and verify that the Properties button appears in the item view command bar.

Summary

In this unit, you learned how to configure command bar buttons and windows shortcut keys using the Configurable User Interface (CUI).

You should now be able to:

- List the components of the Configurable User Interface (CUI) Data Model
- Be able to identify the command bar section locations
- Use a command bar item Action and define Action ranking and security
- Add a Command Bar button and Shortcut.

Review Questions

1. Because the Configurable User Interface provides the ability to reconfigure how users interact with the Aras client, it is strongly recommended that you make a _____ of the database before changing any UI controls that are standard in the Aras interface.

2. In the CUI Data Model, a _____ is associated with the Client, which determines how standard menus and command bars will be organized.
    -
3. How can you control the order of command bar buttons in a toolbar?
    -
4. How can you control who will see a toolbar button?
    -

Lab Exercise

Goal

Add command bar items using the CUI Data Model implementation.

Scenario

A request has been made to make the following adjustments to the client user interface:

- Remove the Change Password in the user menu for Engineering members.
- Add a shortcut key (*alt+h*) which brings up Aras Innovator Help.

Steps

Remove the Change Password in the user menu for Engineering members

1. Open the **Global** Client Presentation for viewing.
2. Set the search mode to Simple on the Command Bar Section tab.
3. Search for location= *MainWindowHeader*. Locate and open for editing the Command Bar Section named **com.aras.innovator.cui_default.mwh_header_user_menu**.
4. In the Command Bar Item tab, notice that the menu button *com.aras.innovator.cui_default.mwh_change_passwd* item already exists in 9th row.
5. Click Add CommandBarItem.
6. Search for the item named **com.aras.innovator.cui_default.mwh_change_passwd**. Click OK.
7. Enter the following:

Sort Order	3800
Action	Remove
For Identity	Engineering

8. Click Done.
9. Logout, then logon again as your personal admin. Notice that the Change Password menu button still appears in the user menu.
10. Logon as Diane Prescott from Engineering ; notice that the Change Password menu button does not disappear in the user menu.

Add a Shortcut Key

1. Open the Global Client Presentation, locate, and edit the Command Bar Section named com.aras.innovator.cui_default.mainViewShortcuts.
2. Add a new Shortcut Item, then fill as follows:

Name	*training.shortcut.Help*
Sort Order	1000
Action	Add
For Identity…	World

3. Right-click and open that Shortcut Item; edit it so it uses the *alt+h* shortcut to call the Handler Method named **cui_default_mwh_onHelpCommand**.
4. Save the changes and close the tabs.
5. Test your work by closing all tabs and then enter the key combination *alt+h*. It may be necessary to logout and login again to see the change.

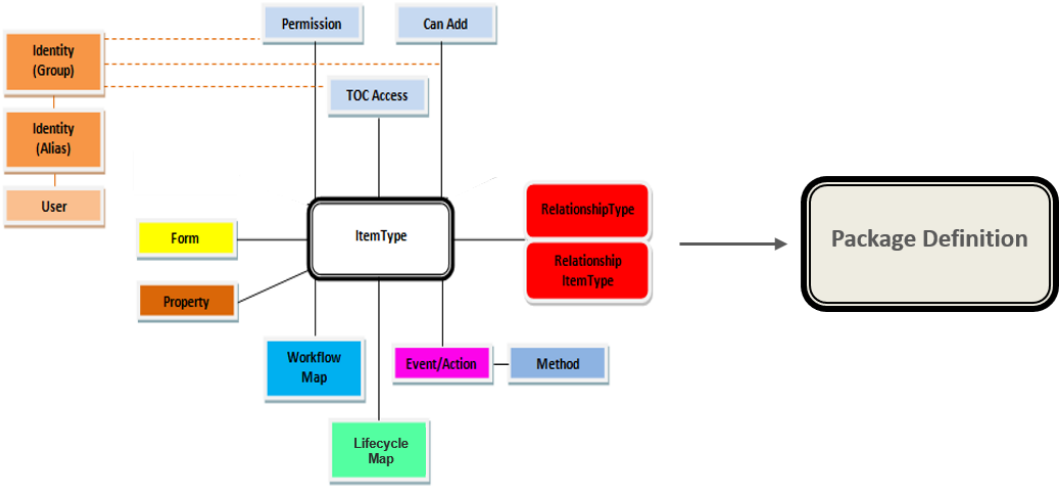
Unit 21 Packaging a Solution

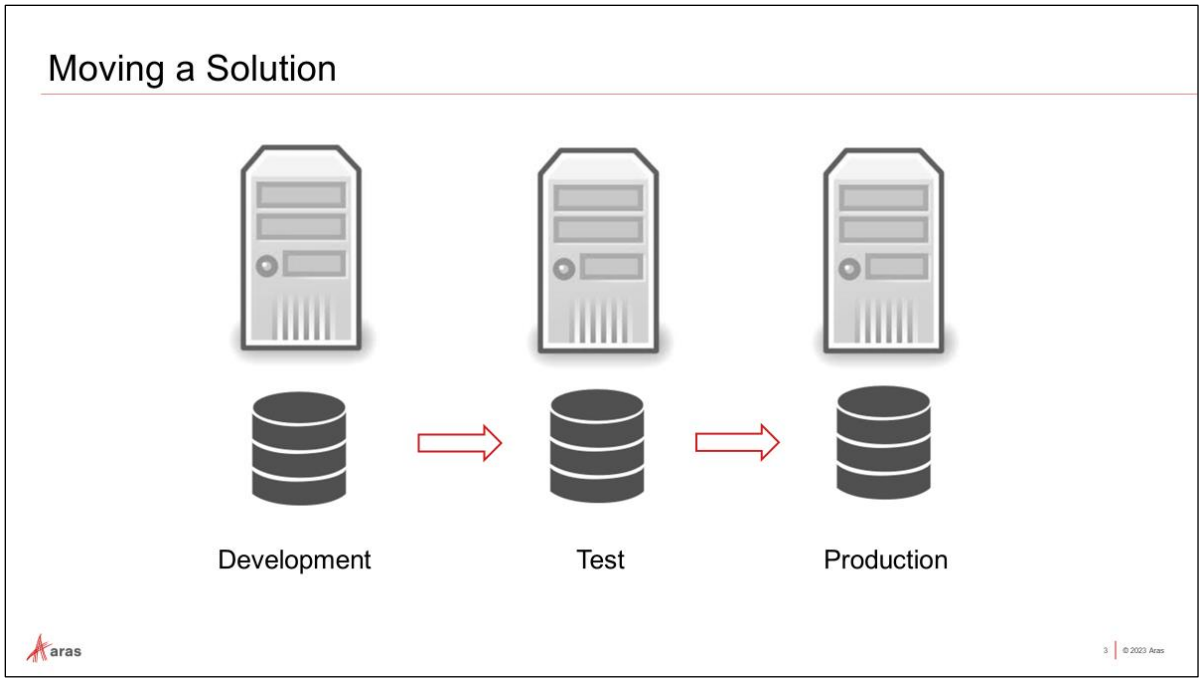
Overview

In this unit, you learn how to package the solutions you built in this course so they can be exported and moved to a different database. You build a package definition, add Items (Elements) to it, and export the package to the file system. You can then import the exported files to a different database.

Objectives

- Identify Package Components
- Create a Package Definition
- Review Package Dependencies
- Add Package Elements
- Review a Package Definition
- Export a Package
- Review the Logs
- Review the Manifest File
- Import a Package.





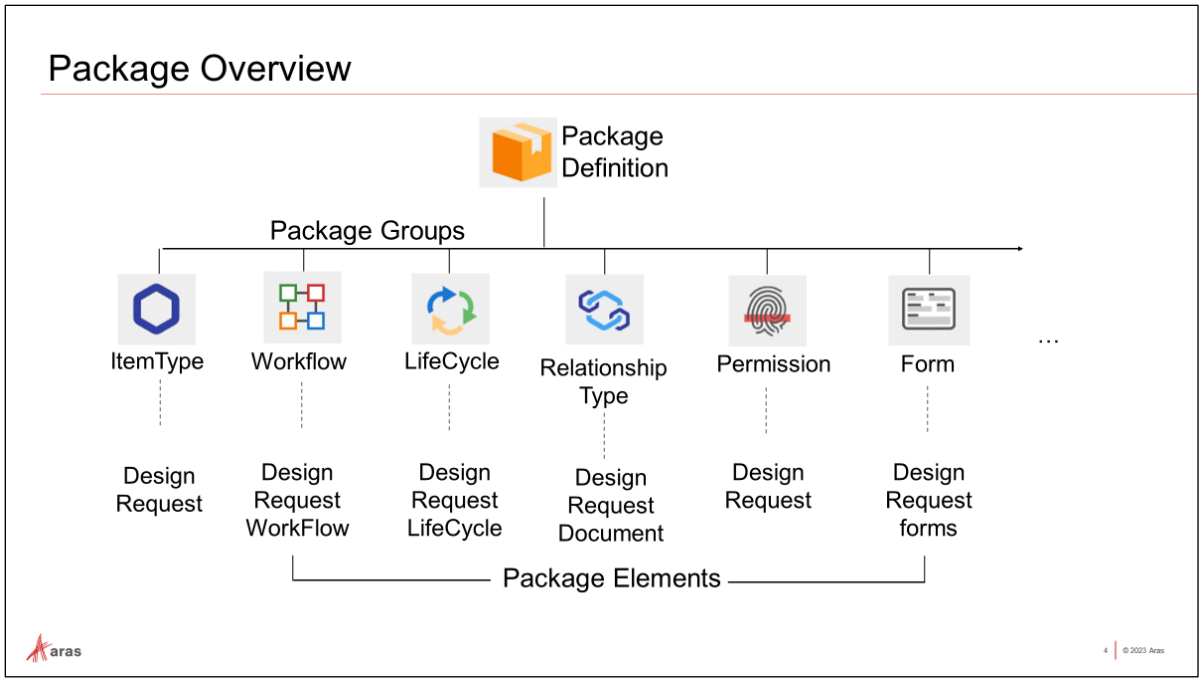
Moving a Solution

When you create a solution, you typically work on a Development server to build and configure the new application. You also use the Development server to correct bugs as well as make enhancements and add new features to the solution. In this course, your cloud server is used as the Development server.

Once a solution is ready for testing, a copy is typically moved to a Test/QA server for assessment of any problems, etc. When the solution is approved for general use, it is then moved to the Production server where end users begin to use the solution. You should never make changes to the Production server until they have been verified through careful testing.

Your exact configurations may differ (and are more detailed), but this general configuration is recommended for developing/deploying solutions.

In this unit, you learn how to package a solution so that it can be transported to a new server. You also learn some best practices for moving solution elements.



Package Overview

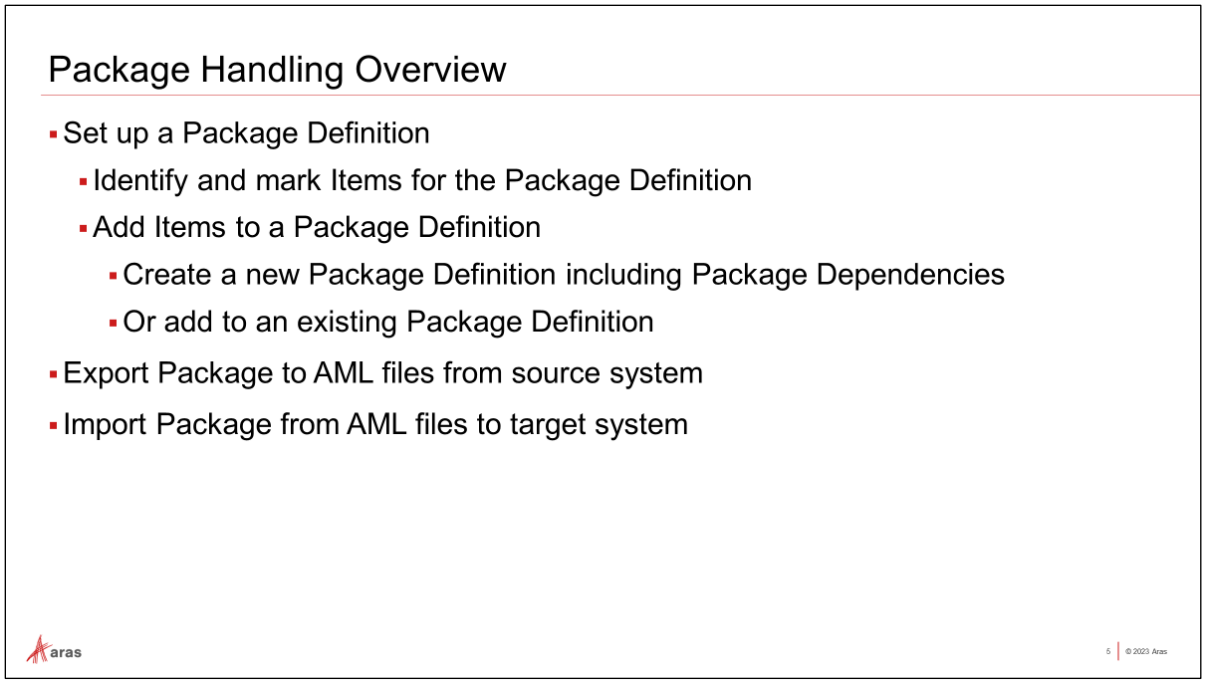
As you learned, everything you create in Aras Innovator is an Item. The templates you created in this course to define ItemTypes, Relationship Types, Forms, etc. are all Items that define your solution.

You also learned that all Items are described using Adaptive Markup Language (AML). In effect, the Aras Innovator database can be represented as a large collection of AML documents that describe the configuration of your solution as well as the data. To move a solution to another system requires careful planning to determine what components comprise your solution.

Package Definition

The first step in moving a solution is to create a Package Definition. A Package Definition is a collection of Package Groups that contain Package Elements. Each Package Group defines a collection of templates that is exported from the database.

In the example above, the Package Definition contains an ItemType Package Group. That group contains one Package Element, which is the Design Request ItemType. Each subsequent group contains respective element(s) for that group (Workflow, Lifecycle map, Relationship Type, etc.).



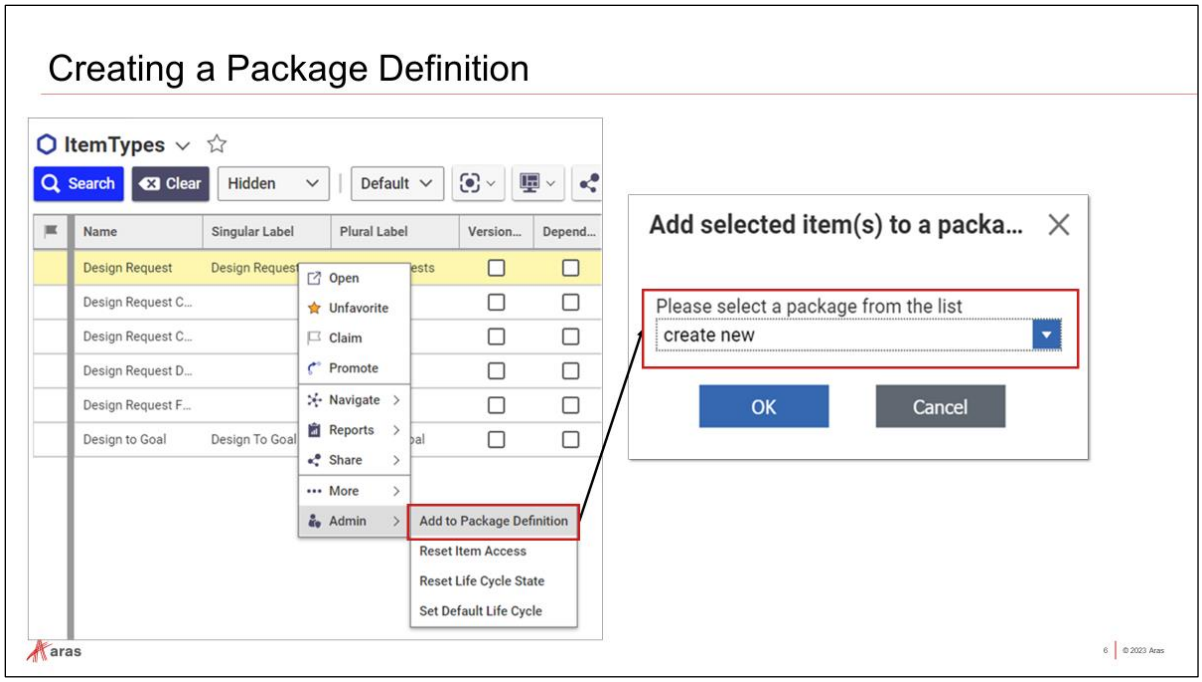
Package Handling Overview

The first step in moving a solution is to set up a Package Definition. The creation of Package Definitions is restricted to Administrators group identity. The starting point are the Items that are to be added to the Package Definition. Right clicking from the Items and selecting Add to Package Definition opens a wizard. This wizard steps you through creating a Package Definition including its dependencies, as well as adding Items to an already existing Package Definition.

Just as everything else in Aras Innovator, the Package Definition itself is also just an Item. Therefore, it can be managed without the wizard. This allows you to modify dependencies if you did not include all needed dependencies on creation of the Package Definition.

The Package Definition is then used by the Export utility. The Export utility program selects data from the source database according to the Package Definition and creates AML files.

The AML files are then available to be imported into the target system. The Import utility adds Items to the target database or modifies existing items, following a control setting of the import dialog.



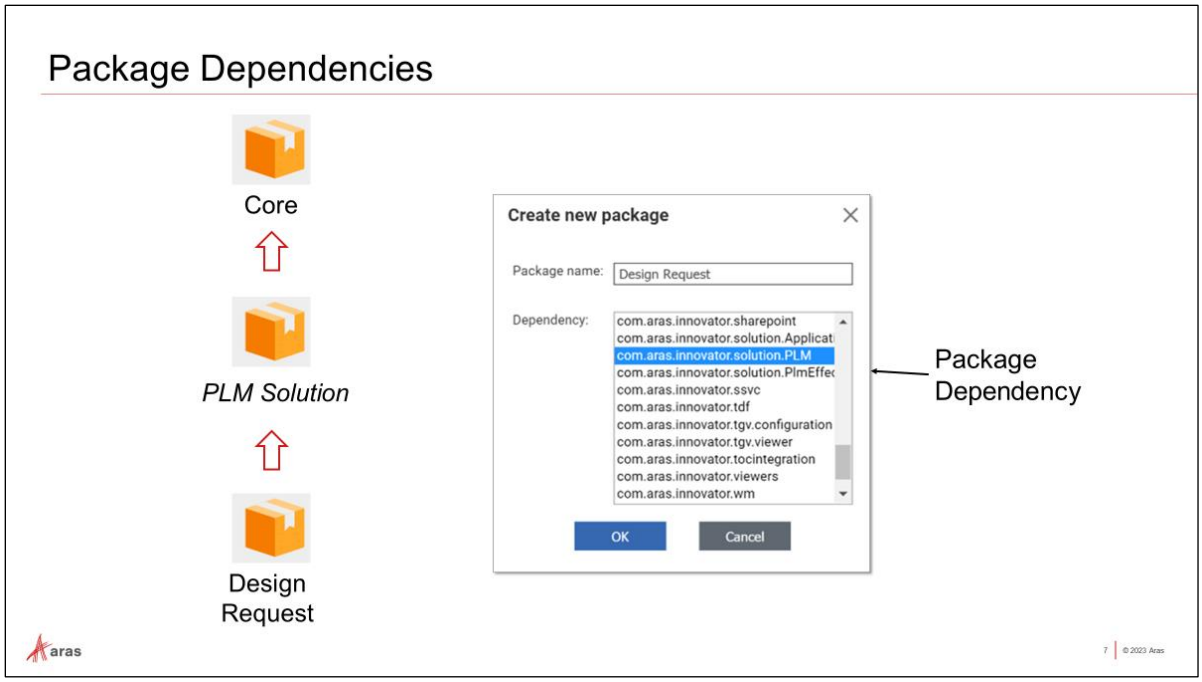
Creating a Package Definition

You create a Package Definition to identify what elements will be exported from the database. You select the elements that make up your solution and add them to a definition. Eventually, the definition is used to export the selected Items from the database to the file system as AML documents.

After creating a package, you can add subsequent elements by selecting each Item and adding them to the same Package Definition.

Try it … Create a Package Definition

1. Navigate to Administration > ItemTypes.
2. Search for the **Design Request** ItemType.
3. Right-click on the Design Request ItemType and choose Admin > Add to Package Definition from the context menu. The Add Item(s) dialog window displays.
4. Select **Create New** from the dropdown list and click OK.
5. Enter *Design Request* for the Package Name.
6. Select the **com.aras.innovator.solution.PLM** package from the Dependency list.
7. Click OK.



Reviewing Package Dependencies

Often, the solutions you build reference elements from other solution packages. You can specify these package references as part of the Package Definition. When the exported package is imported into another system, the required dependencies are resolved before any new elements are accepted into the system. If a required package is missing on the destination system, an error is raised, and the import fails.

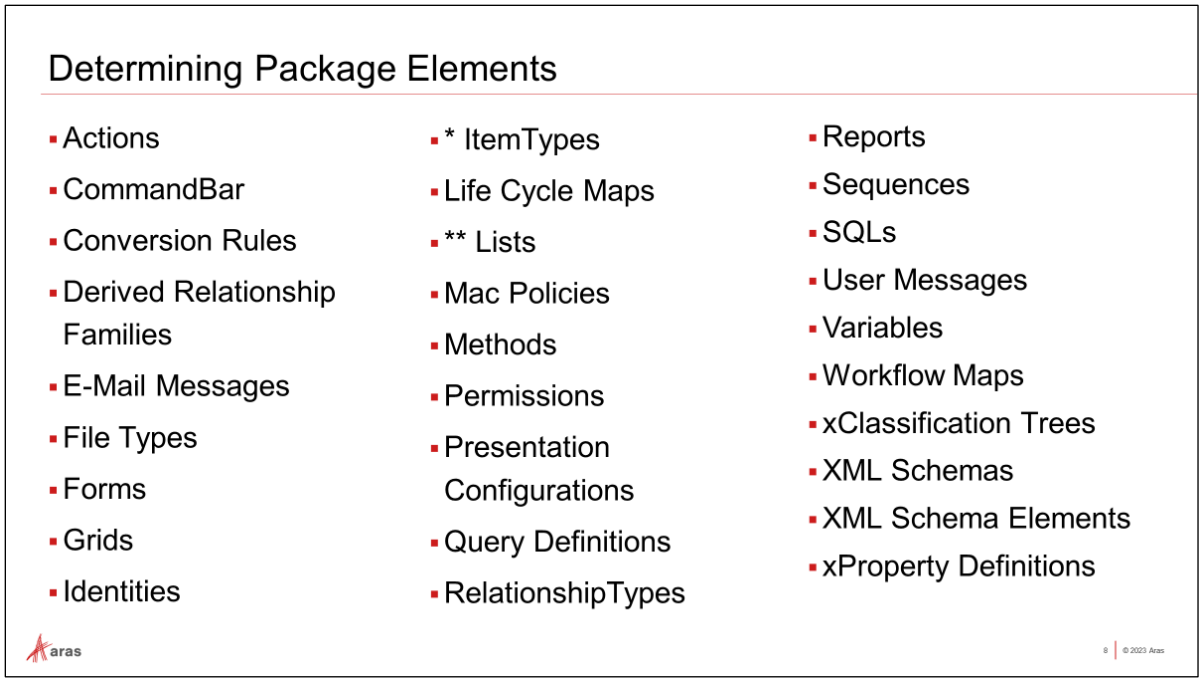
Every solution in Aras Innovator is dependent on the Core Package. The Core Package is installed when you first install Aras Innovator. Because every solution relies on this package, the dependency is implied and does not have to be defined in a Package Definition.

In the example above, the PLM Solution is dependent on Core, but that dependency is implied and does not have to be configured in the PLM Solution Package Definition.

The Design Request solution we built in this course uses elements from the PLM Solution package, e.g., the ItemType Part, so it must be defined in the Design Request Package Definition.

Adding Package Dependencies

While creating the new Package Definition, highlight the Package Definitions that this solution is dependent on and then click the OK button.



Determining Package Elements

Selecting the elements to be included in a package is one of the most important steps in exporting a solution. Because each solution is different, there is no fixed rule on which elements should be selected. Also, remember that each template you create in your solution may reference another element. Some common techniques for collecting the necessary elements include:

- Searching on the modified_on property for each ItemType in question. If you know when development began for your solution you can use this date/time filter to determine what may be relevant.
- Starting with the ItemType(s) that comprise your solution and noting each element that relies on this ItemType (Form, Lifecycle map, Relationship, etc.).
- Using the Where Used browser to determine where an Item is used by other Items.
- Using the Structure Browser to display which Items are related to a parent Item.

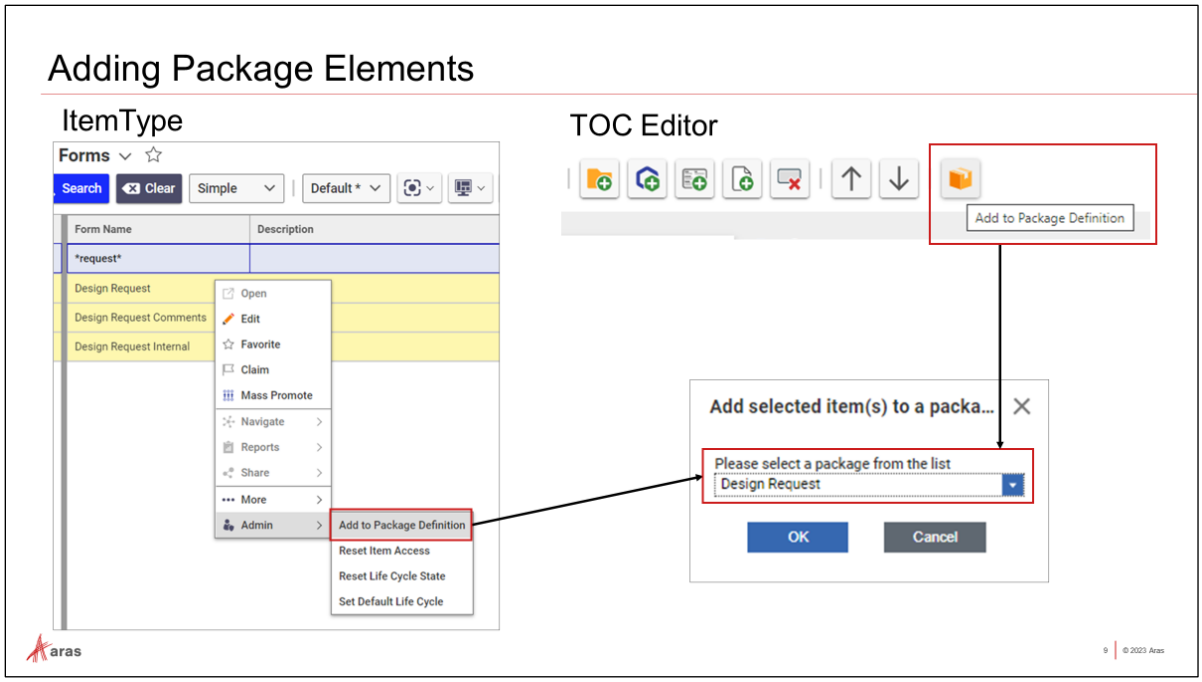
Keeping a complete inventory as you create each Item in your solution can save time when it comes time to move the solution.

* You must not package any ItemTypes that have the Is Relationship option selected (is_relationship=1). These ItemTypes are automatically included in the definition of the RelationshipType element.

** You must not include any Lists that have been generated to bind a set of PolySources to a Poly ItemType. These Lists have the same name as the Poly ItemType and must be newly generated in the destination database.

It is also typically not recommended that you include Users and User Aliases in a Package Definition. Including Users in a package makes your solution less generic and ties your solution to a set of pre-defined individuals that must always exist in the target system.

If you want to transport end user data from one system to another, this is also allowed. For example, just select parts and documents and add them to your package. Packages can be used to have the same test data available at the Test/QA server as at the Development server.



Adding Package Elements

Right-click on the selected Item(s) and choose Add to Package Definition from the context menu. Then, select the desired Package Definition and click the OK button to add the Package Element to the Package Definition.

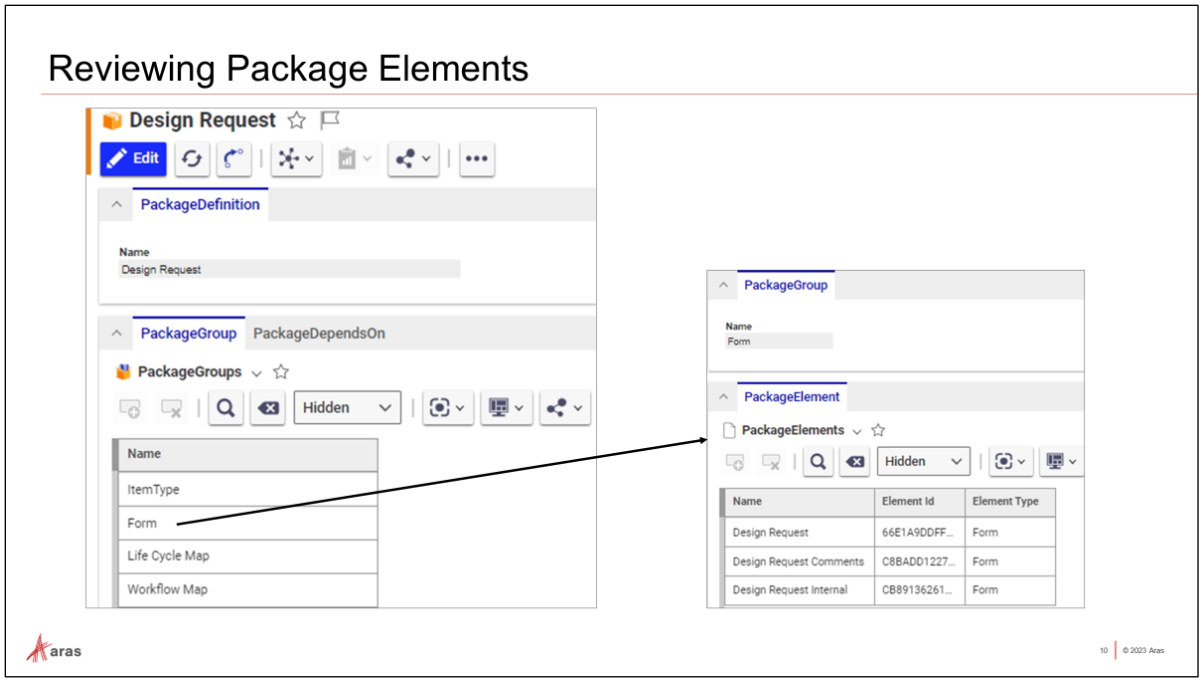
A Package Element can only belong to one Package Definition. If you attempt to add the element to more than one definition, an error occurs.

Package Elements can also be added from the TOC Editor. Select the Element in the TOC Editor and click the Add to Package Definition button.

Try it … Add Package Elements

In this exercise we will add some of the Design Request elements to the Design Request package you created in the previous exercise.

1. Navigate to Administration > Forms and search for all the **Design Request** forms.
2. Select all the Design Request forms, right-click and choose Admin > Add to Package Definition.
3. In the dialog window, use the dropdown list to select the **Design Request** package, then click the OK button.
4. Add the **Design Request** Lifecycle map to the **Design Request package**.
5. Add the **Design Request** and **Calculate Price** Workflow maps to the **Design Request** package.

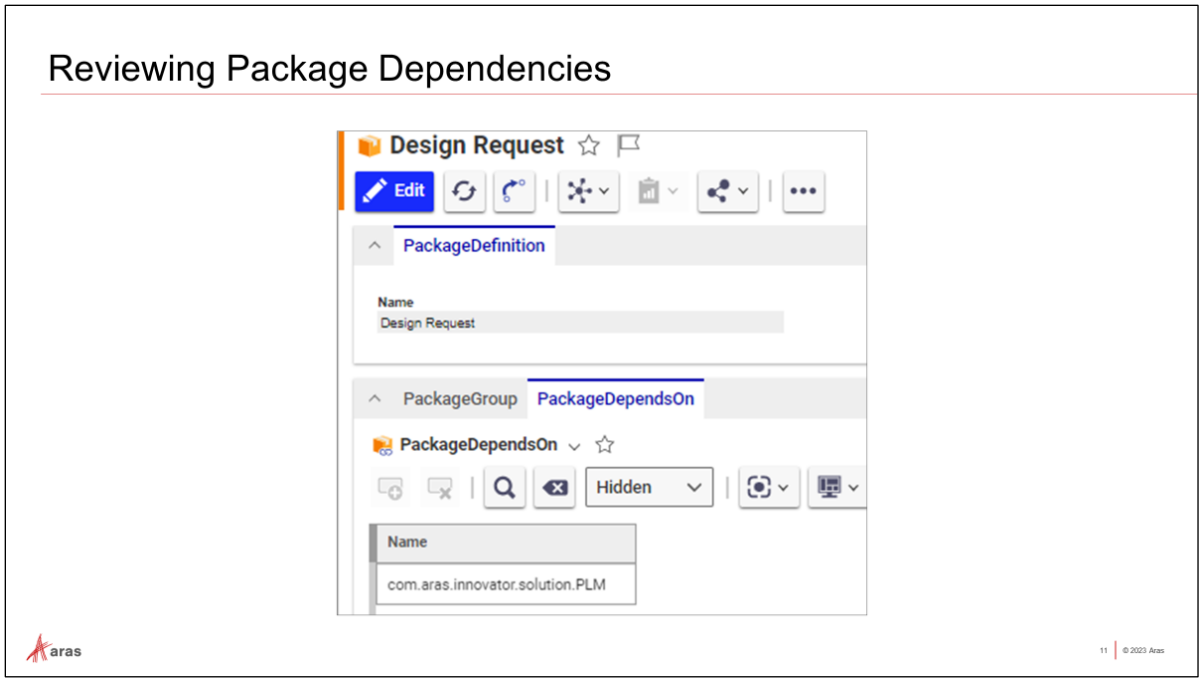


Reviewing Package Elements

Package Definitions can be reviewed and edited. The list of Package Groups displays in the PackageGroup tab in the Relationships grid. In the example shown above, three Form elements are included in the Form Package Group which is part of the Design Request Package Definition.

Try it … Review Package Elements

1. Navigate to Administration > Configuration > Package Definitions and open the **Design Request** package definition for viewing.
2. Right-click on a Package Group and select Package Group > Open from the context menu to see each individual Element Id.
3. Close each open Package Group and return to the Design Request package definition tab.

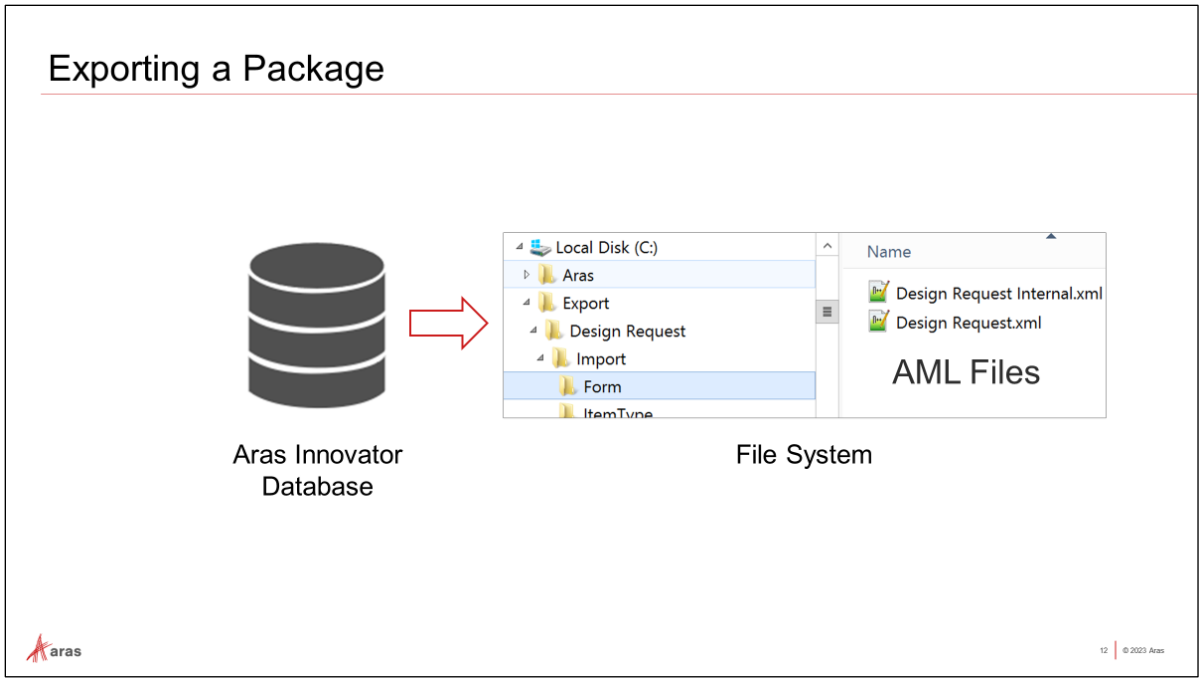


Reviewing Package Dependencies

Dependencies can be added using the PackageDependsOn relationship tab. In the example shown above, one dependency is configured for the Design Request Package Definition.

Try it … Review and add Package Dependencies

1. For the Design Request Package definition, open the PackageDependsOn Relationship tab.
2. If you do not see the **com.aras.innovator.solution.PLM** package definition, click the Edit button, and then click the New PackageDependsOn button.
3. In the name field, enter **com.aras.innovator.solution.PLM** and click Done.

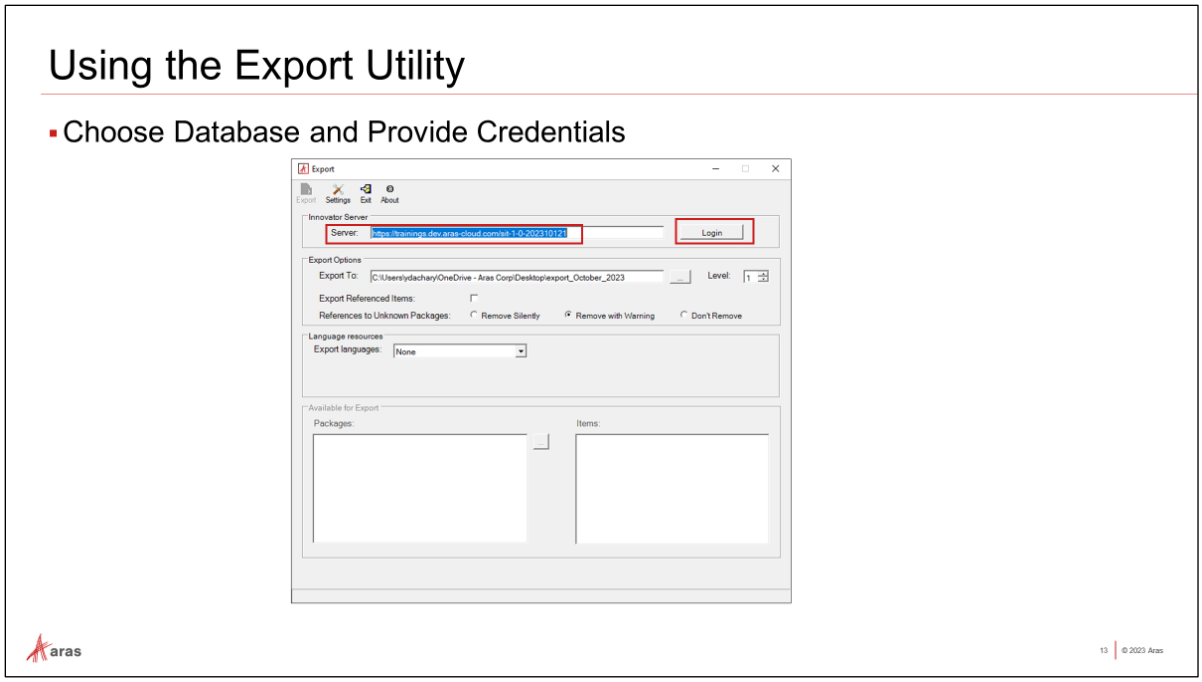


Exporting a Package

Once the Package Definition is complete, the package can be exported using the Export utility. This utility is a separate executable named export.exe that is available on the Aras Innovator CD image. It is important to select the Export utility belonging to the version and service pack of the installation from which you do the export.

The Export utility allows you to select a Package Definition from the database and create a package folder structure in the file system. Each Package Group (ItemType, Form, etc.) becomes a separate subfolder in the file. Within each subfolder, each exported Item is represented as an AML file with the same name as the exported Item.

In the example above, two Form definitions are exported and contained in the Form subdirectory. Note the remaining subfolder names – each represents an exported Item from the database.

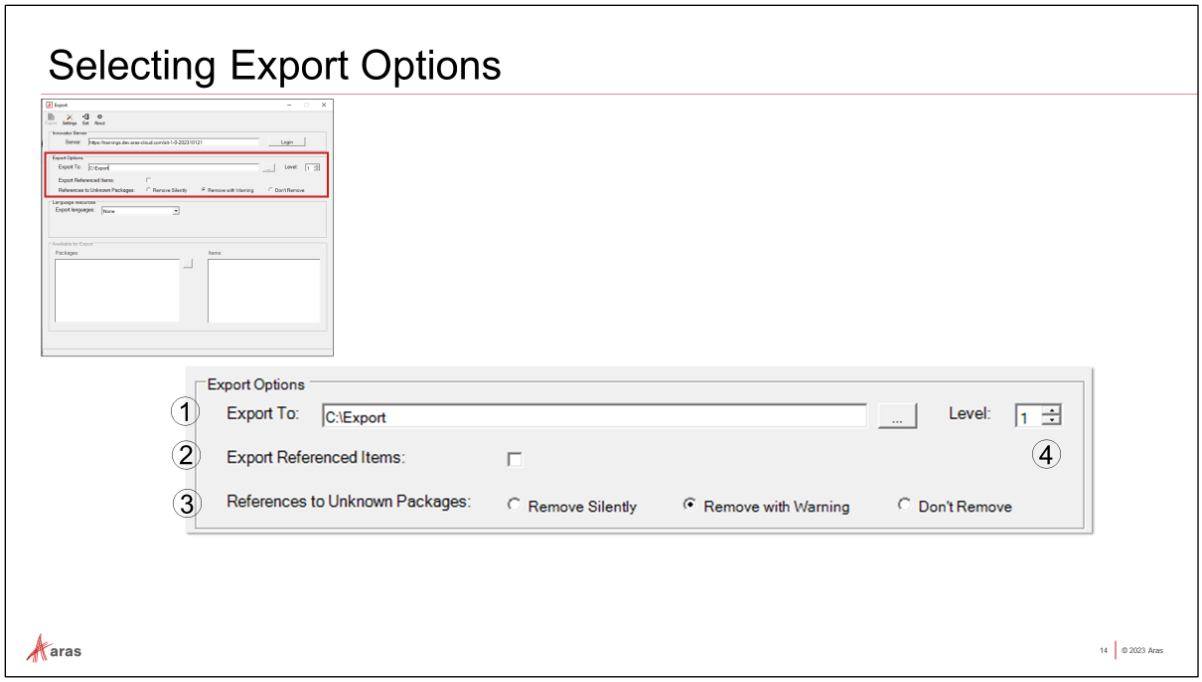


Using the Export Utility

When you run the Export Utility, you must first log on as an administrator using your Aras Innovator username and password to access the selected database.

Try it … Run the Export Utility

1. Use Windows File Explorer and navigate to the place where you have stored the **export.exe** file.
   For example: C:\Program Files (x86)\PackageImportExportUtilities\Export
2. Double click the **export.exe** file to run the Export Utility.
3. In the Server field, enter the URL of the Aras Innovator Server.
4. Click the ellipsis (…) button to the right of the Database field to populate the dropdown list.
5. Select the desired database.
6. Enter a valid administrator username and password in the respective fields.
7. Click the Login button.



Selecting Export Options

The following options can be configured when exporting Items:

① **Export To**

The directory location for the exported AML files.

② **Export Referenced Items**

If a referenced Item in another package is encountered while exporting an Item, this option allows the referenced Item to be exported as well. This is typically not recommended – the Package Definition should be configured with appropriate package dependencies.

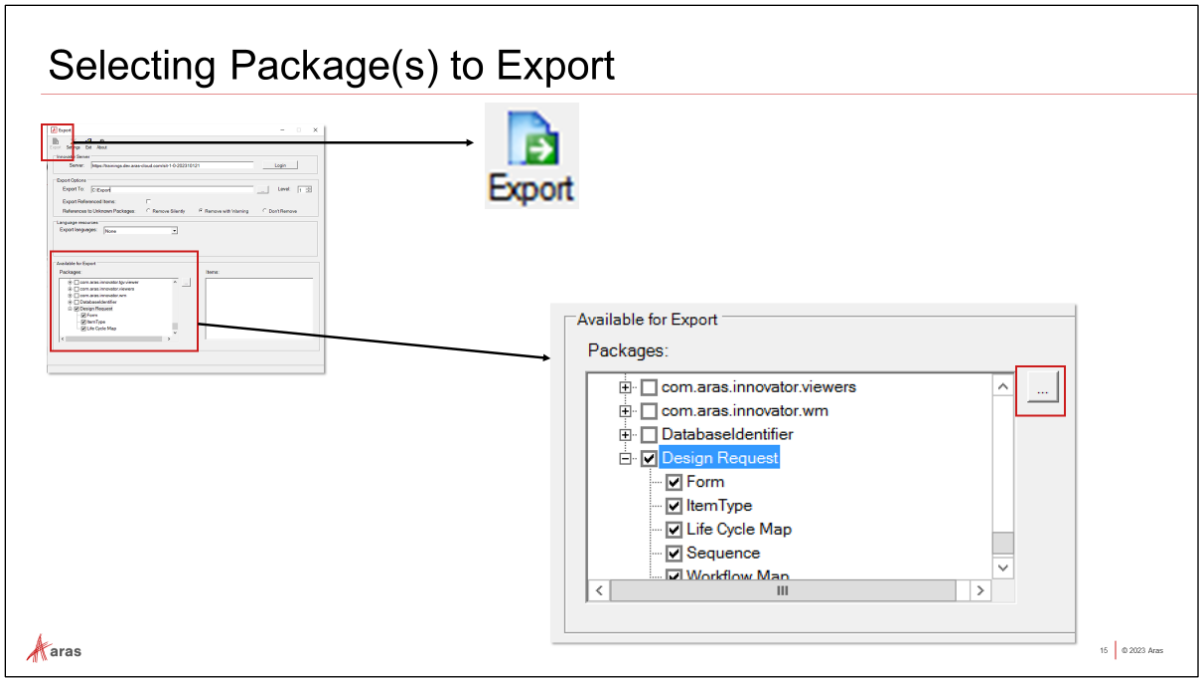
③ **References to Unknown Packages**

Remove Silently – if an unknown referenced Item is encountered when exporting an Item, the Item is removed from the export and no errors occur.
Remove with Warning (recommended) – if an unknown referenced Item is encountered when exporting an Item, the Item is removed from the export and a warning message appears.
Don't Remove – if an unknown referenced element is encountered when exporting an Item, the Item is still exported – this will likely cause problems on Import.

④ **Level**

Uses the AML level attribute to allow a deep query into an Item. Typically, not recommended to modify for basic exports ; use level=2 when exporting a Rpt_Report, to export all data including Rpt_ReportTable Column which is a relationship to rpt_ReportTable.

Referenced Item is used as summary of related Items of Relationship Items as well as Items referenced by an Item Property. Additionally, the term referenced item can include more than one level.



Selecting Package(s) to Export

You can view the Package Elements that will be exported by clicking on the Package Group name. You can then select or deselect individual elements in the Package Group list.

Try it … Select Package(s) to Export

1. For the **Export To** field, select a folder on your local computer.
2. For the Packages field, click the ellipsis (…) button to see the available Package Definitions and their Package Groups.
3. Click the Plus button next to the field your **Database** to expand it.
4. Find and select the checkbox for the **Design Request** package you created.
5. Click the Export button and wait for the export to finish.
6. You can close the Export Utility.

Viewing the Export Log

- Located in the subdirectory *log* of the same directory as the export.exe utility

2022-4-15.13-45.log - Notepad

File Edit Format View Help

```
Login successfully
Login :admin
Server   :https://training.aras-cloud.com/sit-1-0-202204042/
Database :sit-1-0-202204042
         ****Export Design Request Package****
start of execution : 04/15/2022 13:50:42
Item id :74AD28FE87184927B2E473BF2758DECD  keyed_name : Design Request Workflow   type : Workflow Map successfully exported
Item id :F2E6CC8645E7448ABDCC391DCFF4FE0F  keyed_name : Design Request   type : ItemType successfully exported
Item id :C7A4C3FEDF15469A952566A92FB83D1E  keyed_name : Design Request Comments   type : Form successfully exported
Item id :9506CA6677BC46CD81F34A8B72A07C64  keyed_name : Design Request   type : Form successfully exported
Item id :7BEB3A08EEC0415A87C93565844A9B3B  keyed_name : Design Request Sequence   type : Sequence successfully exported
Item id :C4475E84D6A14593A71FFA6E79F73C4A  keyed_name : Design Request Internal   type : Form successfully exported
Item id :DB1093B8B1B64E1CA004DAD8BC5F1CE0  keyed_name : Design Request Lifecycle   type : Life Cycle Map successfully exported
Item id :B409903B527B45038FDDC56F1089523D  keyed_name : Calculate Price   type : Workflow Map successfully exported
```

aras

16 | © 2023 Aras

Viewing the Export Log

A new log file is created for each export session. If you perform more than one export before closing the Export utility, the status messages are appended to the same log file.

Try it … View the Export Log

1. Use Windows File Explorer and navigate to the folder where your Export utility is stored.
   For example: C:\Program Files (x86)\PackageImportExportUtilities\Export.
   A subdirectory named logs exists that stores a log of each export.
2. Open the log file with today's date using a text editor to view results and any errors.
3. Close the text editor when you are finished viewing the log file.

Reviewing the Manifest File

- Contains information for the Import Utility about:
  - Packages to be processed
  - Package dependencies
  - Package disk path locations

```
<imports>
  <package name="Design Request"
           path="Design Request\Import">
    <dependson name="com.aras.innovator.solution.PLM" />
  </package>
</imports>
```

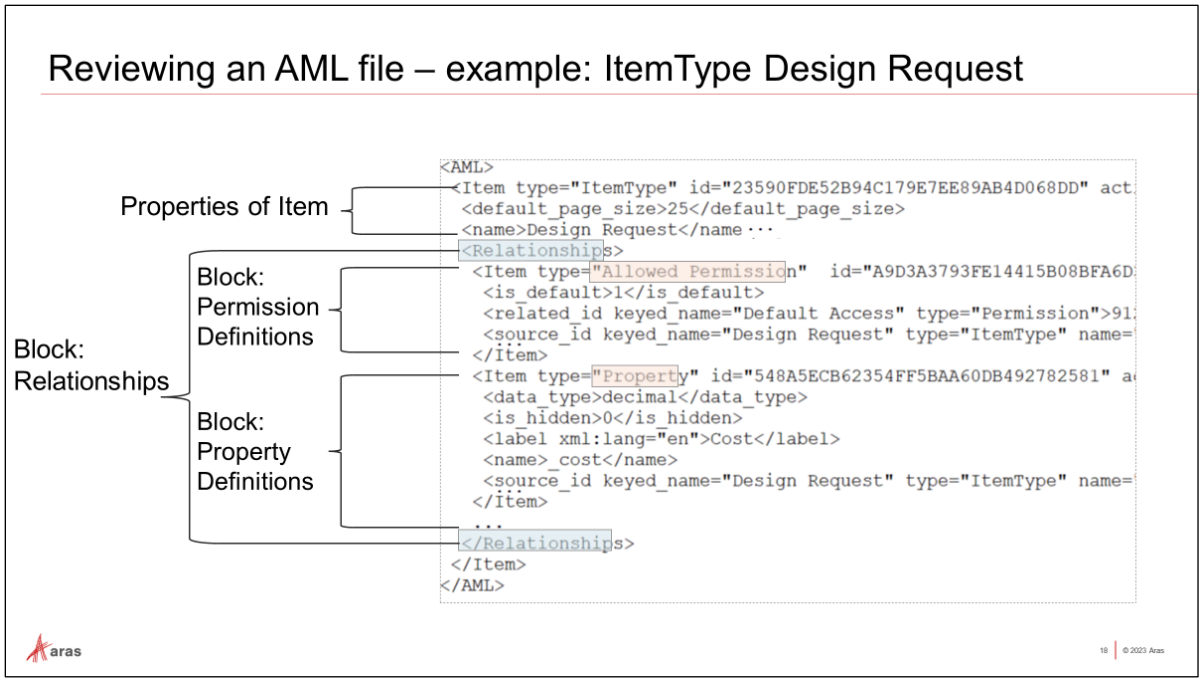
aras

17 | © 2023 Aras

Reviewing the Manifest File

A manifest file is created by the Export utility and is used at Import to locate the package element AML files. The manifest file contains the package name and the top-level directory location where the exported AML files are stored. Each subdirectory under the Import directory is the name of a Package Group (ItemType, Form, RelationshipType, etc.).

The manifest also includes the dependencies that have been configured in the Package Definition so that referenced Items can be resolved on import in the database.



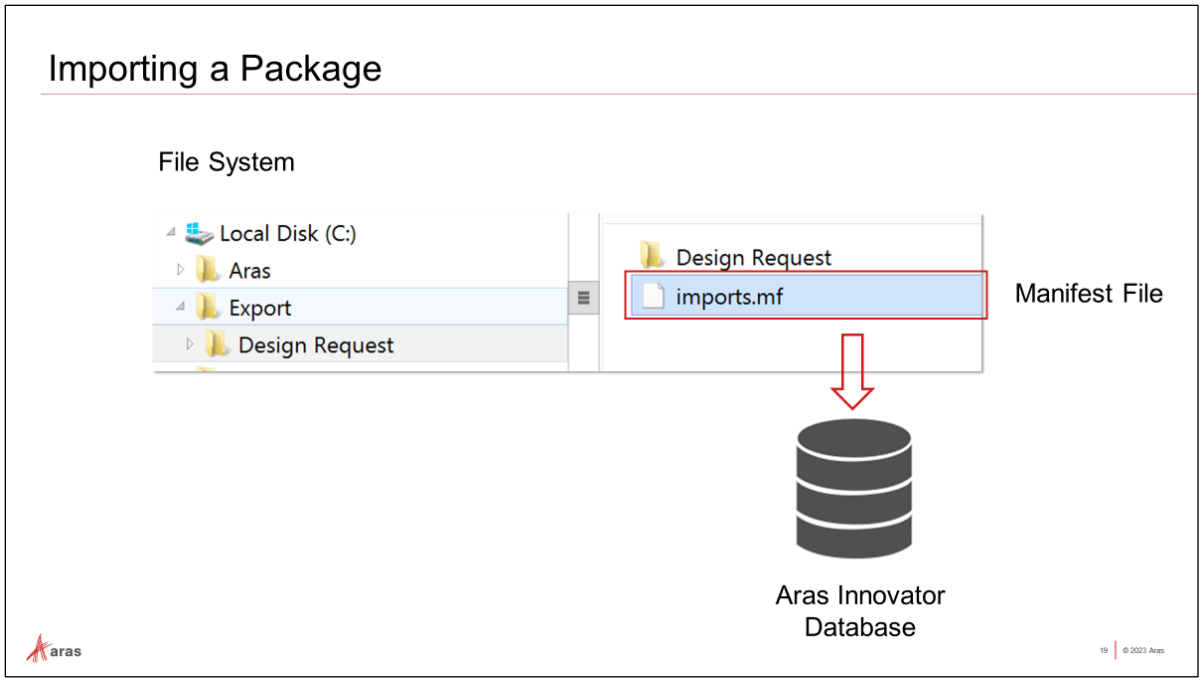
Reviewing an AML file – Example: ItemType Design Request

The Export Utility creates one AML file per Item in the Package Definition. The AML file contains the AML representation of the Item and of all its Relationships. For NULL Relationships this means, that all the relevant information is automatically included in the AML file of the Item. The example above shows the Property _cost belonging to the ItemType Design Request.

In case of Relationships linking to a Related Item, the AML file will contain the Relationship itself, but not the Related Item. The Related Item must be available in the target system before the import of the Source Item AML file.

In the example, a Permission with the name Default Access must already exist in the database before the AML above can be successfully imported. As a result of a successful import, you will not only find the ItemType Design Request, but also the Relationship between the ItemType and the Permission Default Access.

This means that the sequence must be controlled. You can define the control via Dependencies of Package Definitions. The export utility stores the information about the import sequence in the dependson tag in the Manifest file.



Importing a Package

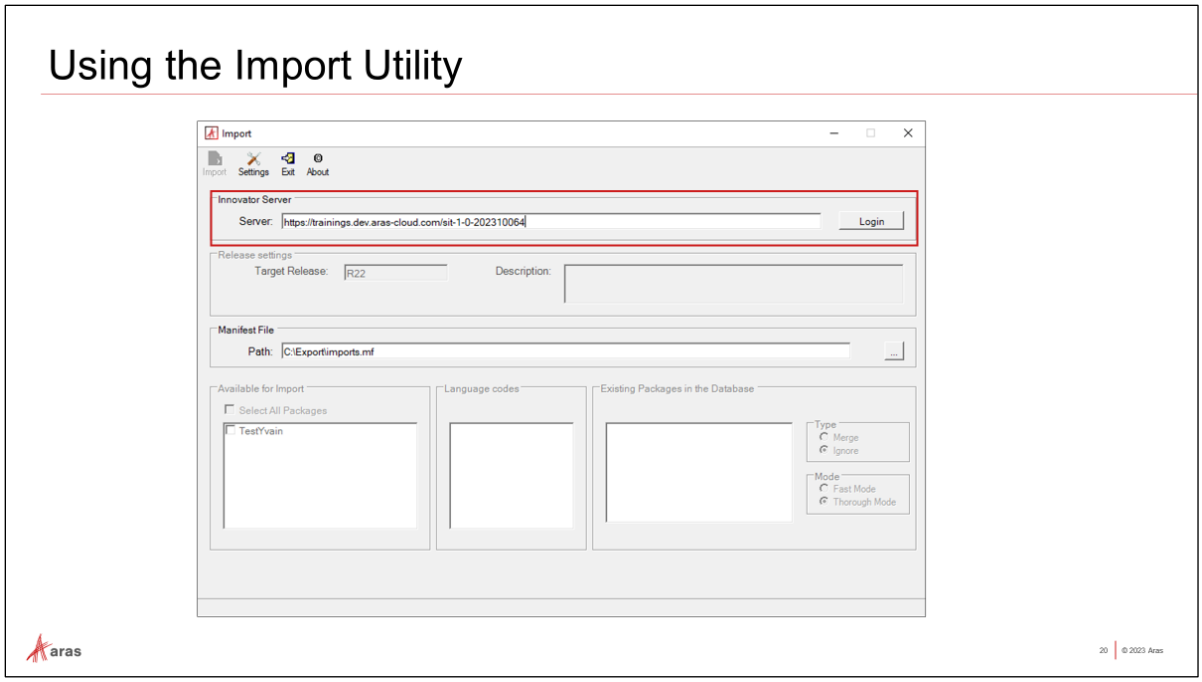
Package import is accomplished by using a separate executable named Import.exe that is available on the Aras Innovator CD image. It is important to select the Import utility belonging to the version and service pack of the installation to which you do the import.

The Import utility uses an existing manifest file to locate the AML export files. Each file is then parsed to create (or edit) new Items in the destination database.

The Import Utility imports Package Groups from a Package Definition in the following order:

1. List	11. Email Message	20. Field
2. Sequence	12. Action	21. Property
3. Revision	13. Report	22. View
4. Variable	14. Form	23. SQL
5. Identity	15. Workflow Map	24. Metric
6. Member	16. Lifecycle Map	25. Chart
7. User	17. Grid	26. Dashboard
8. Permission	18. ItemType	
9. Method	19. RelationshipType	

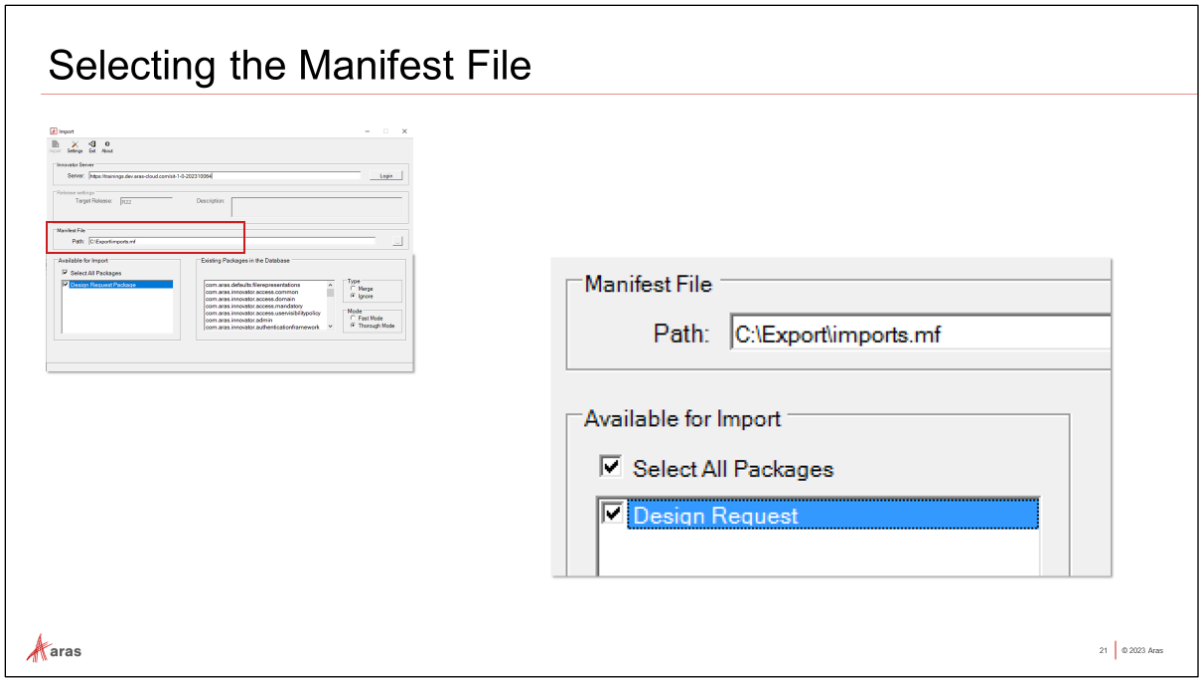
If this sequence is not supporting you well enough, please use different packages and define the control using dependencies.



Using the Import Utility

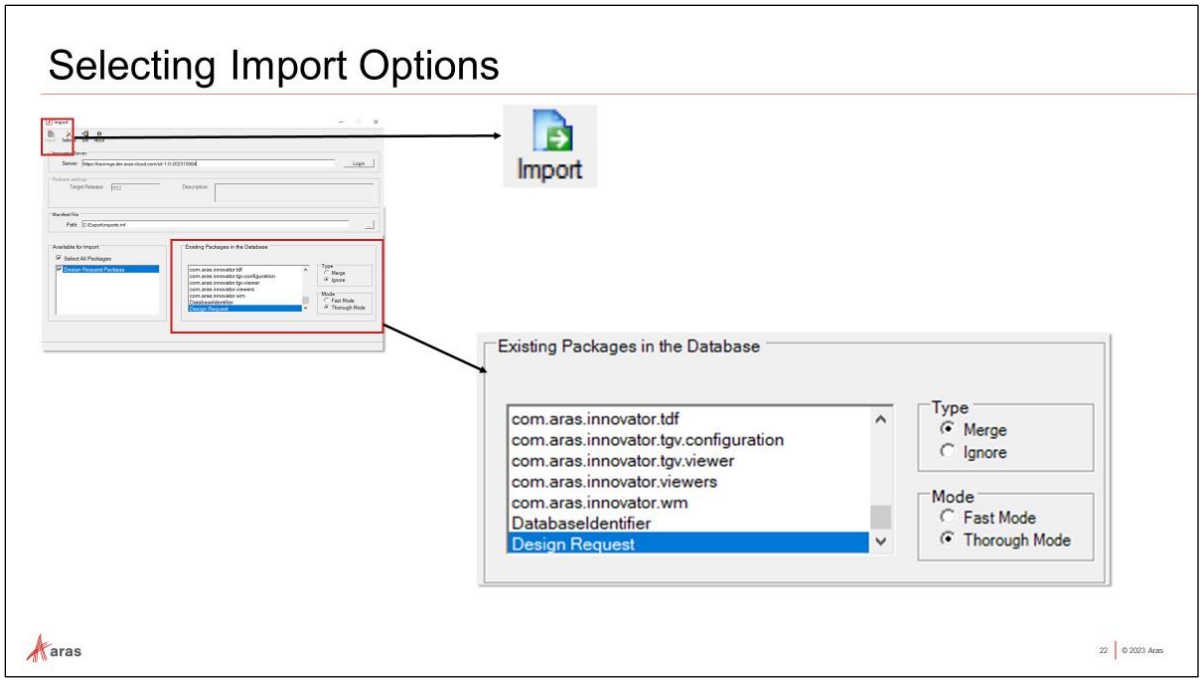
Run the Import Utility and enter the Aras Innovator URL in the Server field. Then click the ellipsis (…) button and select the desired destination database from the dropdown list. Enter a valid username and password for the destination database.

Note: when importing a rpt_Report, Aras Innovator will set the user who performs import as the report creator. This modifies the Creator compared to the original DB.



Selecting the Manifest File

Provide the path to the desired manifest (.mf) file and select the Package for import.



Selecting Import Options

Type

Merge –attempts to merge changes of an imported Item into the database if the same Item already exists (recommended).

Ignore –ignores the Item if it already exists in the database.

Mode

Fast Mode – will not do a detailed reference checking of the Item as it is imported into the database.

Thorough – performs comprehensive checking of referenced Items to ensure they exist and are correct (recommended).

Running the Import Utility

Switch the Type from Ignore to Merge, then click the Import button (in the upper left corner).

Summary

In this unit, you learned how to create a Package Definition to export your solution for transport to another server. You should now be able to:

- Identify the Package Components
- Create a Package Definition
- Add Package Elements
- Review a Package Definition
- Export a Package
- Review the Logs and Manifest File
- Import a Package

Review Questions

1. Why are Package Dependencies important when exporting a solution?

2. Where does the Package Definition get stored?

3. What is the format of the exported Item files?

4. When is a manifest file used?

Lab Exercise

Goal

Export the Change Request solution using a Package Definition and the Export utility.

Scenario

In this exercise, you will create a Package Definition for the Change Request solution. You will need to determine all the required components for the solution to create a new Package Definition. You will then export the solution.

Steps

Relationship Types
Forms
Lifecycle Maps
Workflow Maps
Lists
Sequences

Identities
Permissions

1. Create a new Package Definition named Change Request and capture all the elements you have listed in the previous step.
2. Use the Export utility to extract the elements from the database to the file system into a directory named C:\ExportPackage.
3. Review the log and manifests files that are created and resolve any errors.

List of Exercises and Labs

Unit 1 Introduction 5
Unit 2 Users and Identities 13
Try it … Create a New User 16
Try it … Create Engineering Users 16
Try it … Open the Alias Identity of your personal User Item 18
Try it … Add a User to a Group 20
Try it … Modify a Group Identity 21
Try it … Run an Identity Membership Report 24
Lab Exercise 26
Unit 3 Exploring ItemTypes 27
Try it … Open an Existing Item Type Definition for Viewing 29
Try it … View ItemType Properties 33
Try it … Run an ItemType Report 36
Try it … Create the Design Request ItemType 39
Try it … Provide Access Permissions to the Design Request ItemType 40
Try it … Create a Class Structure 41
Lab Exercise 43
Unit 4 The Table of Contents (TOC) Editor 45
Try it … Create a Category 51
Try it … Move a Category 51
Try it … Create a Child Category 51
Try it … Add an ItemType button to a Category 52
Try it … View the TOC using the All Employees Identity 53
Try it … Customize an ItemType button in the TOC 54
Lab Exercise 56
Unit 5 Itemtype Properties 57
Try it … Modify some properties settings 68
Try it … Make a Property Required, and Modify its Label 68
Try it … Create Custom Properties for the Design Request ItemType 72
Try it … Explore the impact of search settings on the search grid 73

Try it … Open existing saved searches 74
Try it … Open the Design Request Main Search Grid to View Properties 75
Lab Exercise 77
Unit 6 Creating Object Properties 79
Try it … Create a Foreign Property 82
Try it … Create a List and use it for a new Property 83
Try it … Create a Filtered List 84
Try it … Create List and Filter List Properties 85
Try it … Create a Sequence 87
Try it … Defining an Image Property 88
Lab Exercise 90
Unit 7 Creating the User Interface 93
Try it … Regenerate a Form 96
Try it … Edit the Design Request Form 98
Try it … Configure Additional Properties on the Form 99
Try it … Add Unused Properties to the Form 101
Try it … Copy a Form 106
Try it … Create and Classify a View 108
Try it … Create a Responsive Form 111
Try it … Add Properties on the canvas 115
Try it … Change the Columns and Group box settings 117
Try it … Create a Design Request 118
Lab Exercise 120
Unit 8 Creating Item Relationships 123
Try it … View the RelationshipType Search Grid 125
Try it … Set Default Structure View to Tabs On 128
Try it … Create a RelationshipType 129
Try it…Require a Related Item 132
Try it … Create a Design Request Document Relationship 134
Try it … Create a NULL Relationship 136
Try it … Test the NULL Relationship 137
Lab Exercise 141
Unit 9 Assigning Security 143
Try it … Create a New Permission 147
Try it … Assign a new default Permission 147
Try it … Change Access to an Item 150

Try it … Configure an ItemType to Allow Private Permissions 151
Try it … Set a Private Permission 152
Try it … Create a New Team 155
Try it … Assign Team Permissions 156
Try it … Display the Team Property on the Design Request Form and Test Team Permissions 157
Try it … View or Change Password Settings 160
Lab Exercise 162
Unit 10 Visualizing Data 165
Try it … Create a New Query Definition 170
Try it … Select Properties 171
Try it … Execute a Query 172
Try it … Add an item from an Item Property and add Properties 173
Try it … Add items from a Relationships tab and include properties 173
Try it … Add Referenced Items and include properties 174
Try it … Create a Tree Grid View 176
Try it … Add Columns/Labels 177
Try it … Map Elements 178
Try it … Combine Rows 179
Try it … Change the Icon 179
Try it … Configure Cell Display Settings 181
Try it … Create Item Hyperlinks 182
Try it … Set TGV Usage 183
Try it … Review the TGV Result 184
Try it … Define a Query Parameter 185
Try it … Map a Query Element to a Condition 185
Try it … Enable a Tree Grid View Parameter 186
Try it … View the Reuse Query Element Definition 187
Lab Exercise 189
Unit 11 Dashboards and Reporting 193
Try it … Explore the cui_Dashboard permissions 195
Try it … Create a new Dashboard 196
Try it … Add a Tree Grid widget 198
Try it … Add a Report Widget 199
Try it … Create and share saved search and use it on new widgets 199
Try it … Change the widgets localization on the Dashboard 201

Try it … Create a Report 203
Try it … Define aggregations and groups on a Report 206
Try it … Export the Report 206
Try it … Manage Filter and Sharing with on a Report 207
Try it … Create a report table in a Dashboard 209
Try it … Create a Column Chart 210
Try it … Create a Widget Template 213
Lab Exercise 216
Unit 12 Defining Extended Classification 221
Try it … Grant Classification Administrator Privileges 226
Try it … Create Extended Properties 229
Try it … Edit an xClassification Tree 231
Try it … Assign xProperties 232
Try it … Change Access for Item Classification Permissions 234
Try it … Change Access for an xProperty Value Permission 235
Try it … Assign Extended Classification Permissions 236
Try it … Classify an Item 238
Try it … Allow Private xProperty Permissions 239
Try it … Assign a Private xProperty Permission 239
Try it… Test Private xProperty Permissions 240
Lab Exercise 242
Unit 13 Defining Lifecycle Maps and Versioning 245
Try it … Create and Configure an ItemType Lifecycle Map 247
Try it … Set Specific State Settings 249
Try it … Configure a Permission on a Lifecycle State 250
Try it … Create Transitions and Provide Roles 251
Try it … Associate an ItemType with a Lifecycle Map 252
Try it … Create a New Item and View the Lifecycle Map 253
Try it … Promote a Design Request to Incomplete 254
Try it … Investigate ItemType Versioning 256
Try it … Create a Major Revision Sequence 257
Try it … View a Version 258
Try it … Set the Behavior for a Relationship Type 261
Try it … Set the Lifecycle State Behavior 262
Lab Exercise 268

Unit 14 Creating a Workflow Map 271
Try it … Create a Workflow Map 272
Try it … Insert New Activities 273
Try it … Add Activity Paths 274
Try it … Add a Break 275
Try it … Set Start and End Activities 279
Try it … Define an Automatic Activity 280
Try it … Create Assignments for all Activities 282
Try it … Create Tasks within a Activity 286
Try it … Create an Activity Variable 290
Try it … Set Default Path after Automatic Activity 293
Try it … Set an Override Path with Authentication 294
Try it … Assign a workflow to an ItemType 296
Try it … Test the Workflow 297
Lab Exercise 299
Unit 15 Enhancing a Workflow 301
Try it … Change the Lifecycle Map 304
Try it … Configure a New Lifecycle Promotion 305
Try it … Test the Workflow 306
Try it … Create the Workflow 309
Try it … Configure a Workflow to Call a Subflow 311
Try it … Configure the ItemType and test 312
Try it … Configure the Managed by Property 313
Try it … Change Assignments 314
Lab Exercise 317
Unit 16 Working with Files 319
Try it … Define a Relationship for the Container and a File Item 331
Try it … Modify the RelationshipType 331
Try it … Modify the Relationship ItemType 332
Try it … Test the Relationship ItemType 332
Try it … Create a New Property 333
Try it … Add the Property to a Form 333
Try it … Test the Item Property File Relationship 334
Lab Exercise 336
Unit 17 Replicating Files 337
Try it … View the available Vaults 341

Try it … Configure Read Vault Priority 343
Try it … Create a Replication Rule 345
Try it … Define the Replication Rule settings 347
Try it … Specify File Types 348
Try it … Specify a Target Vault 349
Try it … View Transactions in Process 350
Try it … View Completed or Failed Transactions 351
Lab Exercise 360
Unit 18 Sending Notifications 361
Try it … Create a Notification Message 365
Try it … Create an E-Mail Message with variables - Lifecycle 370
Try it … Configure a Lifecycle E-Mail Message 372
Try it … Use Variables in a Workflow Notification 373
Try it … Configure a Workflow Notification 375
Lab Exercise 377
Unit 19 Implementing Poly Items 379
Try it … View a Poly Item ItemType 381
Try it … Select a Poly Item 383
Try it … Create a Poly Item ItemType 384
Try it … Assign Common Properties 385
Try it … Add the Catalog Items Poly Item to the TOC 386
Try it … Query a Poly Item 387
Try it … Create a Relationship to a Poly Item 388
Try it … Create an Item Data Type Property using a Poly Item 389
Lab Exercise 391
Unit 20 Configuring Command Bars 393
Try It … Open the Global Presentation 401
Try It … Open a Command Bar Section 403
Try It … Add a Button Command Bar Item 403
Try It … Configure the Button Command Bar Item 404
Try It … Configure an ItemType-specific Command Bar section 405
Try It … Configure the Button Command Bar Item 406
Try It … Open Command Bar Section 408
Try It … Add a Shortcut Command Bar Item 409
Try It … Configure a Shortcut Command Bar Item 411

Lab Exercise 413
Unit 21 Packaging a Solution 415
Try it … Create a Package Definition 419
Try it … Add Package Elements 422
Try it … Review Package Elements 423
Try it … Review and add Package Dependencies 424
Try it … Run the Export Utility 426
Try it … Select Package(s) to Export 428
Try it … View the Export Log 429
Lab Exercise 437

This page intentionally left blank

Appendix A Review and Lab Questions Answer Key

Unit 2 – Users and Identities

Review Questions

1. An Alias Identity. User privileges are defined and managed through identities.
   An alias identity is unique for each user.
   Additional rights and access will be inherited from groups or roles that the alias identities belong to.
2. The Identity Membership report is the best way to see a Group Identity hierarchy.
3. Disable logon.

Unit 3 – Exploring ItemTypes

Review Questions

1. ItemTypes are templates which define properties, forms and views, Lifecycles, Workflows, permissions, relationships and server, and client methods. Items are instances created from an ItemType template.
2. Minimum requirements are: Name, Can Add Identity, and Default Permissions. TOC Access should also be considered.
3. Allows the Aras Administrator to select one or more properties to identify an object in searches, tab titles, reports, etc. that is more descriptive and user friendly than the object GUID, or numbered Items.
4. ItemType Definition Report.
5. Aras Administrator did not grant TOC Access to their identity or a group that they belong to.
6. Shared properties can be defined at the parent level and inherited by all subclasses.
   Subclasses can then be differentiated by additional properties, a different Lifecycle, a different view of a form, or different permissions for who can create a new item.

Unit 4 – The TOC Editor

Review Questions

1. Categories and ItemType buttons.
2. Use the TOC Editor's "View As" feature.

Unit 5 – Creating Properties

Review Questions

1. In the SQL table named PROPERTY.
2. Values are stored in rows in a table created for each Item Type.

3. If Hidden is checked, the property is not visible or searchable in the main search grid using a Simple Search. If unchecked, the property is visible as a column in the main search grid and can be included in simple searches. If Hidden2 is checked, it is not visible on a relationship grid tab. If unchecked, the property appears as a column on a relationship tab.
4. Properties maintain a label that can be changed, and it is used for displaying in search grids and form views.
5. System Properties – 23 properties automatically created and needed for each new ItemType by the Aras Innovator System.
   Custom Properties –Properties for one ItemType, defined by an administrator to store data specific to the ItemType.

Lab Exercises

Follow the Lab instructions.

Unit 6 – Creating Object Properties

Review Questions

1. False, Foreign properties are populated from the values of existing properties of a different item. They are displayed as read only when referenced on the form.
2. Select or create an Item Property and use this as the Data Source.
3. An administrator can edit a Sequence and set or change the current value.

Lab Exercises

Follow the Lab instructions.

Unit 7 – Creating the User Interface

Review Questions

1. A form establishes the main user interface for adding, viewing, and editing items. A view is the relationship between an ItemType and a Form.
2. There is an unlimited number of forms that can be linked to an ItemType. However, each form must be linked through the view relationship tab on the ItemType.
3. If you have already built and saved a customized form. This action rebuilds the form, placing unhidden propertied on the form.
4. The width and height can be set on the Form Properties tab of the Form Editor.
5. A responsive Form adapts to the user' screen size, all properties are managed in same location, the visual grouping of fields allows a clearer fields layout, the tab order is embedded.

Lab Exercises

Follow the Lab instructions.

Unit 8 – Creating Item Relationships

Review Questions

1. RelationshipType Item defines the connections or links to other items. This can control the number of items allowed to be attached and versioning behavior. A Relationship ItemType stores the properties about the specific link, such as Parent item, child, quantity, etc.
2. A Relationship ItemType.

3. A Null relationship allows the ability to add multiple rows of information of a similar type to an item such as remarks, comments, etc. without connecting to another DB Table.

Lab Exercises

Follow the Lab instructions.

Unit 9 – Assigning Security

Review Questions

1. Yes, with a private Permission. This private Permission is specific to one item only and is never shared; it does not appear in the ItemType's Permissions tab.
2. Grant Can Discover but not Get privilege on the permissions for the items created from an ItemType template. Remark: if you grant "Show Permissions Warning", the user will receive a warning message about the lacking access.
3. Navigate to Administration > Identities, open the context menu, and select Identity Membership. Alternatively, open any Identity (such as All Employees). Click the Reports button and select Identity Membership to run the Report.
4. Navigate to Administration > Variables, edit the "AccountLockoutThreshold_triesNum" variable, and set the value to 3.

Lab Exercises

Follow the Lab instructions.

Unit 10 – Visualizing Data (QB and TGV)

Review Questions

1. Relationships, Referencing Item.
2. XML or AML.
3. Tree Grid View.
4. Map table cells.
5. Create Parameters in the Query Definition and set visibility for Parameters in Tree Grid View.
6. Open the Cell Display Settings editor and set the Cell View Type to Item, fill the additional fields of ItemType Name and Item Property. Use the GUID of the target Item in the Item Property. Prerequisite: The id is contained in the Query result.

Lab Exercises

Follow the Lab instructions.

Unit 11 – Dashboards and Reporting

Review Questions

1. Dashboard Authors and Dashboard Administrators.
2. A report can be used:
   - on its own, to display data with Preview
   - to create a report table widget in a dashboard
   - to create a report chart widget in a dashboard

3. Help the widget creators by pre-configuring widgets templates with restrictive choices for specific purposes.

Lab Exercises

Follow the Lab instructions.

Unit 12 – Defining Extended Classification

Review Questions

1. Property reuse, class grouping, enhanced security.
2. Add the user identity to the Classification Administrators group Identity.
3. Create and maintain xProperties, xClasses, and xPermissions – assign xClassification Trees to ItemTypes and xPermissions to combinations of xClassification Trees with ItemTypes.
4. Property Value Permission, Item Classification Permission
5. xp-

Lab Exercises

Follow the Lab instructions.

Unit 13 – Defining Lifecycles and Versioning

Review Questions

1. When you use a change process to edit Items in a Released state, the Major Rev property is incremented to next value from the Revisions sequence.
2. Transitions define the role/identity that can promote the item to the next state.
3. Item behavior is controlled both by the RelationshipType and of the Item behavior on the Lifecycle state.
4. The standard Lifecycle behavior takes precedence over the RelationshipType behavior setting, foreseen on the RelationshipType with the setting as Float or Fixed. With Hard Float or Hard Fixed the Relationship Type behavior takes precedence.
5. Use the Navigate > Versions menu. Or you can use the Structure Browser and click the Side-by-Side with Other Version button.

Lab Exercises

Follow the Lab instructions.

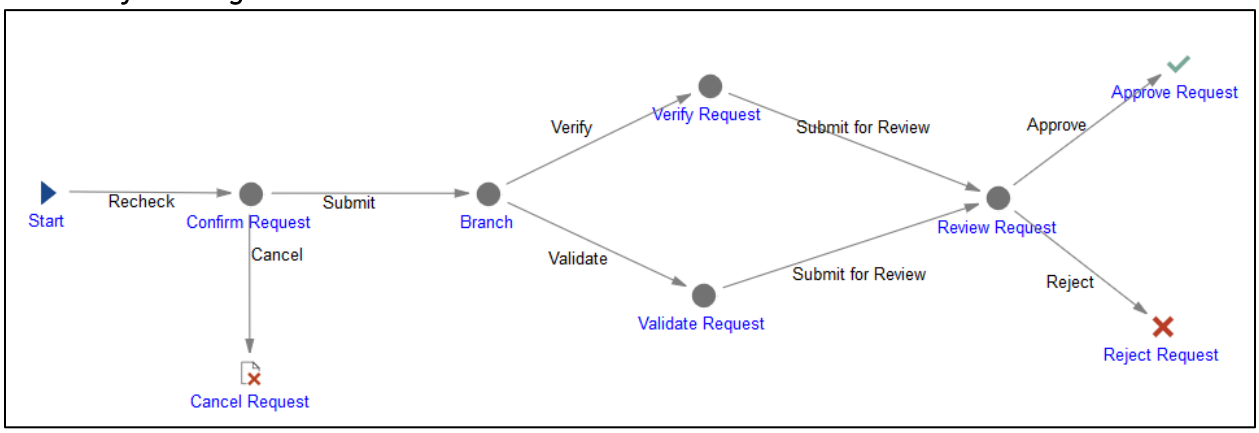
Unit 14 – Creating a Workflow Map

Review Questions

1. A Workflow Map graphically represents the template for a business process. It usually contains activities, paths, assignments, and tasks. A Workflow Process is an instance created using the Workflow Map, tracking actual participants and which assignments are completed.
2. From the assignments tab, select the desired Group Identity and then check *For All Members*.
3. Create an automatic activity and then create multiple default paths to link to other activities (branching out).
4. One or more Default Path(s) must follow automatic activity.
5. Password and E-signature.

Lab Exercises

Lab Workflow Diagram:



Follow the Lab instructions.

To enable the Branch activity to process both the Verify Request and Validate Request activities in parallel: set the Branch activity as Automatic activity and ensure that both exit paths are Default paths.

To ensure that the Review Request Activity is not reached until the Verify Request and Validate Request activities have been completed: check 'Wait For All Inputs'.

Unit 15 – Enhancing a Workflow

Review Questions

1. A Workflow promotion bypasses any identity role defined in the Lifecycle transition. The identity on the assignment tab for the workflow activity assumes role.
2. The Automatic Activity setting must be set to true on the node launching the subflow.
3. Configure as a Role Assignment so that only a member of that role can be assigned.

Lab Exercises

Follow the Lab instructions.

Unit 16 – Working with Files

Review Questions

1. False, properties about files are stored in the SQL database, but the files are stored in a secure folder structure on the vault server. Exception: Icon files are stored on the Aras Innovator Server for purpose of fast access.
2. The Aras Innovator Administrator assigns a default vault when creating or modifying a user.
3. The File ItemType defines the patterns to recognize files to configure view control.
4. Option 2. Add a property to the ItemType for the file container with Data Type = Item and the Data Source = File items. After saving, the property must be displayed on the form(s).

Lab Exercises

Follow the Lab instructions.

Unit 17 – Replicating Files

Review Questions

1. The User's Default Vault.
2. Add a replication rule to the vault item for where the file was originally vaulted. Add a relationship to one or more vaults on the Target Vaults tab of the Replication Rule.
3. Edit the Replication Rule Form and select the FileTypes tab. Add a new relationship for the specific file type (i.e. AutoCAD).
4. Edit the User's profile and set on the Read Priority tab.

Lab Exercises

Follow the Lab instructions.

Unit 18 – Sending Notifications

Review Questions

1. User Notifications are received by the user in the Aras Innovator client application, and are seen either from the Notifications button (top right of screen) or in a Pop-Up upon login or after a defined unit of time, while e-Mail Notifications really send emails to the users' mail address.
2. An administrator configures a User Notification in Administration > Notification > Notification Messages.
3. E-Mail Notifications can be generated from Lifecycle Transitions, Lifecycle States, Workflow Activities via Visual Collaboration and programattically through a Method.
4. The Aras Innovator Server must be configured to work with a valid SMTP e-Mail server.
5. Any Property associated with an ItemType can be inserted into an e-Mail message as a variable to be substituted with the actual value from the database table.
6. Edit the Workflow Map and select the Activity to invoke the e-Mail message. Select the Notifications tab and add a new notification. Set the event to *On Delegate*.

Lab Exercises

Follow the Lab instructions.

Unit 19 – Implementing Poly Items

Review Questions

1. The Implementation Type of the ItemType must be set to Poly Item. Then, use the PolySources relationship tab to select the desired ItemTypes.
2. When consistent Properties are used in multiple ItemTypes and a single view is needed to search for an item. When the related item of a relationship can be represented by more than one ItemType. When a Property of Data Type Item may be represented by more than one ItemType.
3. RelationshipTypes, Views, Workflows, Permissions, Reports, Actions, Client Events, Secure Social Collaboration and xProperties.
4. When the relationship can support different types of Items (multiple ItemTypes).

Lab Exercises

Follow the Lab instructions.

Unit 20 – Configurable User Interface

Review Questions

1. Be sure to make a BACKUP copy of the database before changing the UI in Aras.
2. Presentation Configuration
3. By the Sort Order of the Command Bar Item inside a Command Bar Section. Please be aware that the ItemType Client Style Presentation can overlap command bar sections, menu entries, toolbar buttons or shortcut keys and thus will finally decide about the presentation of the buttons..
4. Command Bar Items can be qualified by an Identity as well as Item Classification.

Lab Exercises

Follow the Lab instructions.

Unit 21 – Packaging a Solution

Review Questions

1. Most solutions reference elements from other solutions. On import, required dependencies must be resolved before any new elements are accepted. If the required package is missing on the new destination, the import will fail with an error condition.
2. The Package Definition is an item that can be found under the Administration category in the TOC. Select Administration > Configuration > PackageDefinition and search for the desired definition.
3. AML documents.
4. A manifest file is used at Import to locate the package element AML files.

Lab Exercises

Follow the Lab instructions.

If you have followed along with the naming suggestions in the labs, you can filter searches by naming patterns or use dates that items were created or modified on.

The following table lists the items that were created or edited in the Change Request lab assignments this week. Be sure to add all of them to your package definition before using the Export utilities.

ItemTypes	Change Request
Relationship Types	Change Request Customer, Change Request Memo, Change Request Product
Forms	Change Request, Change Request Defect
Lifecycle Maps	Change Request
Workflow Maps	Change Request, Quality Control
Actions	None were created this week.
Methods	None were created this week.
Lists	Regions, Design Centers
Sequences	Change Request
Identities	All Employees, Engineering
Permissions	Change Request, In Validation Request