ProApp Designer

Release 14 - Package 22

Last Modified: 04/06/2026

Copyright Information

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Phone: 978-691-8900

Table of contents

Aras Background and Cron Jobs User Guide

Introduction

Purpose

This User Guide describes how to use Aras Background & Cron Jobs application for external users.

Scope

This document provides instructions for configurations and functionality of Aras Background & Cron Jobs application.

Target Audience

This document is intended for external users with limited access to data.

Guide Organization

This document describes how to use the Aras Background & Cron Jobs application. This User Guide is organized into the following sections:

SECTION 1: **Introduction**	This section defines the purpose and scope of the User Guide, identifies its intended audience. It also supplies a brief overview of each section to help users in navigating the guide.
SECTION 2: **Introduction to Background & Cron Jobs**	This section explains the purpose of the Background & Cron Jobs feature and offers an overview of the functionality available within Aras Background & Cron Jobs.

Introduction to Background & Cron Jobs

Background Jobs enable users to execute heavy, time-consuming operations such as custom actions using server methods in the background, ensuring smoother performance and user experience. Cron Jobs allow administrators to schedule custom actions at defined intervals through a simple UI that supports recurrence configuration. Administrators can also define queues, which can be tailored to manage timeouts, cutoffs, and pre/post-processing scenarios. The Queue Dashboard provides real-time visibility into the status of all background jobs, including tools for failure remediation. Upon job completion, email notifications are sent to inform users of the job status.

Role

When the Background & Cron Jobs package is deployed, it automatically loads the CRON JOB Administrator group identity by default. This identity must be assigned to all users who will be managing or interacting with the jobs.

Typical Workflow

Background Jobs can be created by following below steps:

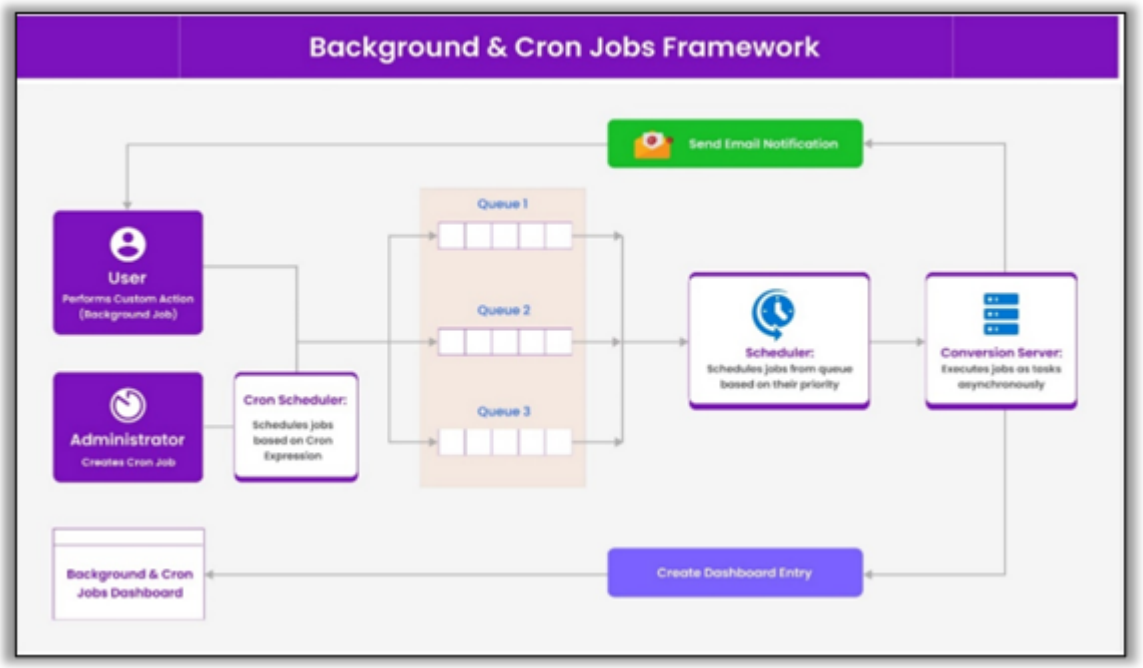
- Administrators can create custom Actions on any ItemType or by using any server method, enabling the creation of jobs.
- Created, jobs are executed in the background within a designated queue.
- Upon job completion, the output and status are automatically communicated to the user via email notification.

Cron Jobs can be created by following below steps:

- User should login as “Admin”, then Go to TOC → Administrators → Pro App Designer → Background & Cron Jobs → Cron Jobs.
- Click on “Create New Cron Job” select all the mandatory fields.
- Attach created Cron Job to the relationship of Queue & it will execute as per defined schedule.

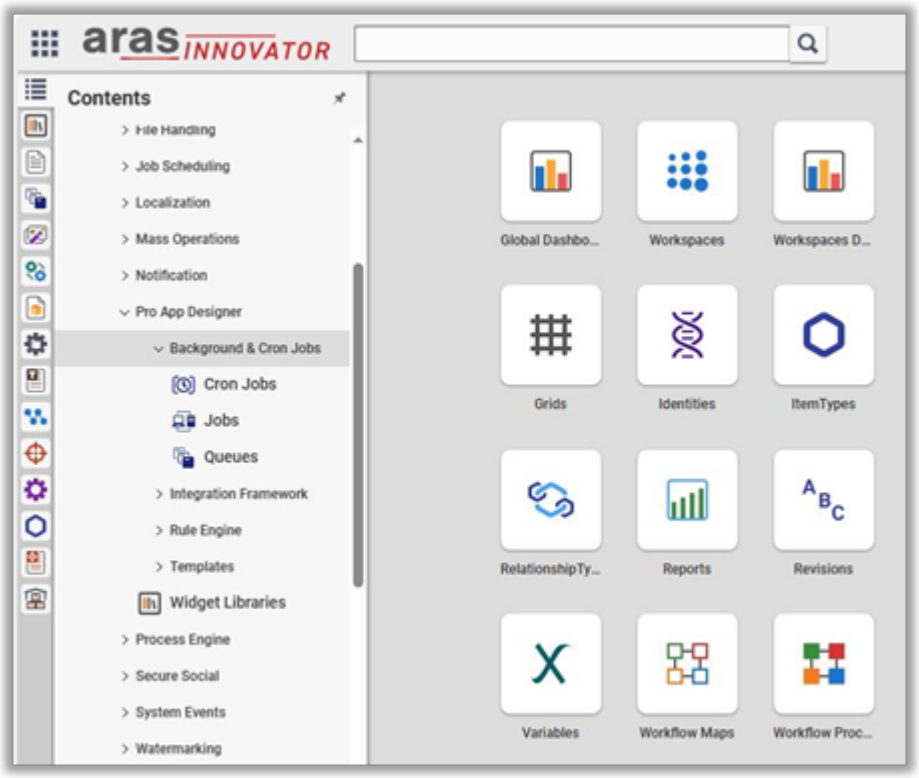
Architecture

The Background & Cron Jobs Framework architecture is primarily composed of three key components **Queue**, **Scheduler**, and **Conversion Server**. When a user or administrator manually initiates or executes a custom action, the corresponding background or cron job is added to the **Queue** for processing. The **Scheduler** then evaluates and prioritizes these jobs before scheduling them for execution. Once scheduled, the **Conversion Server** carries out the jobs asynchronously, ensuring minimal disruption to system performance. Upon completion, the job results are displayed on the dashboard, and an email notification is sent to the relevant user.



TOC

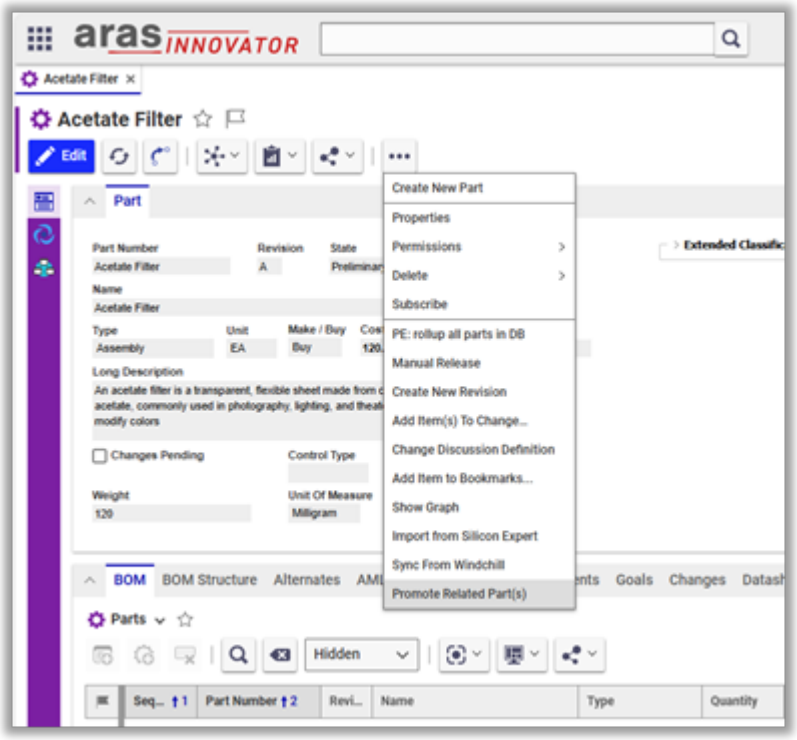
Once Background & Cron Jobs Framework package is installed, you will get see a category called **“Background & Cron Jobs”** under TOC à Administrators à Pro App Designer. This category shows underneath Cron Jobs, Jobs and Queues.



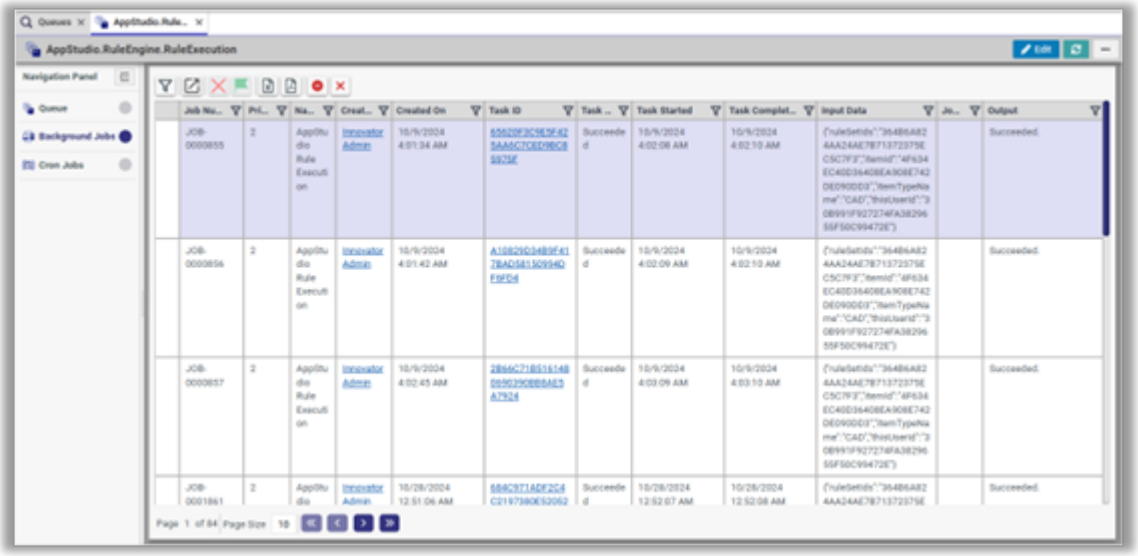
Jobs

Background Job

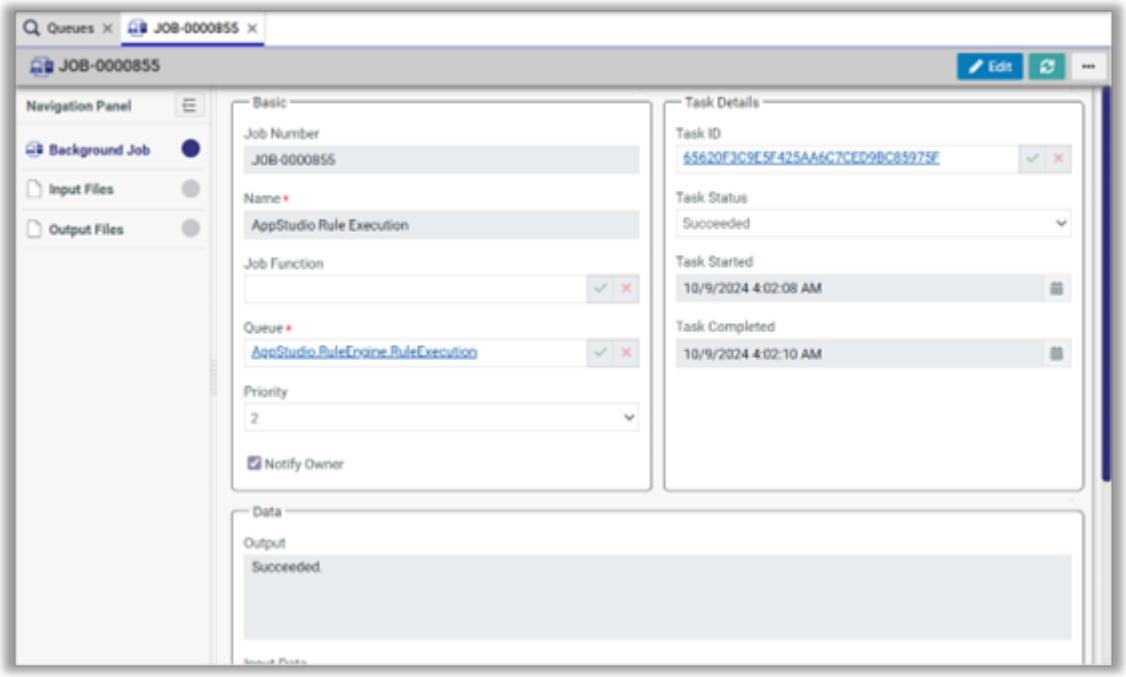
- Admin can **create** an **Action** on any ItemType or can use any server method to create the Jobs.



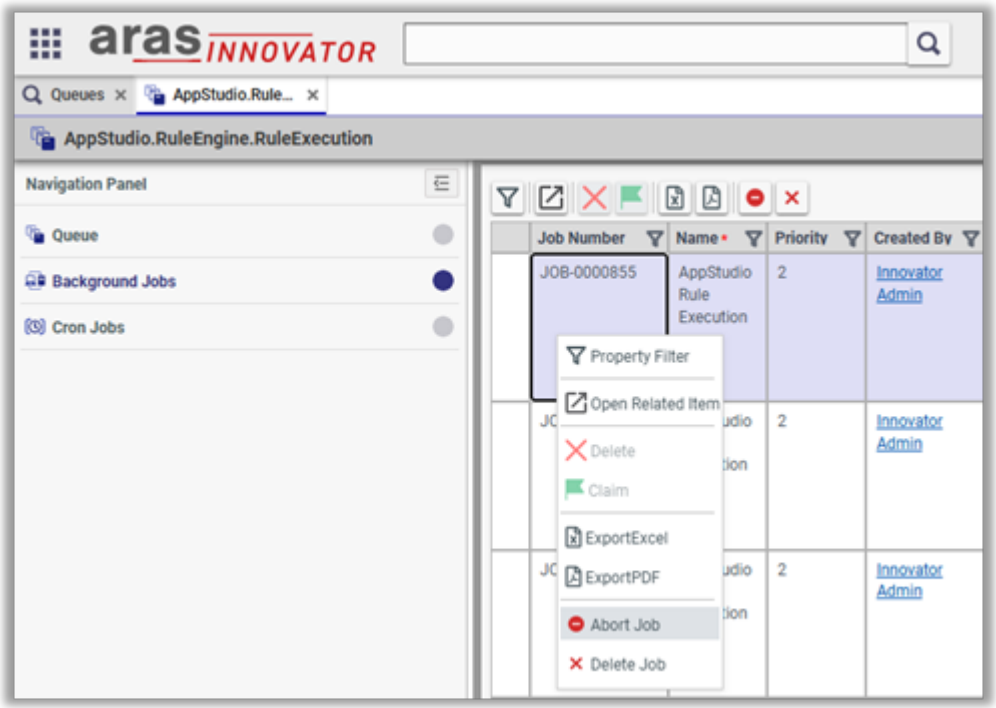
- For example, when user clicks on the Action **“Promote Related Part(s)”**, a new Job is created and added to the queue.



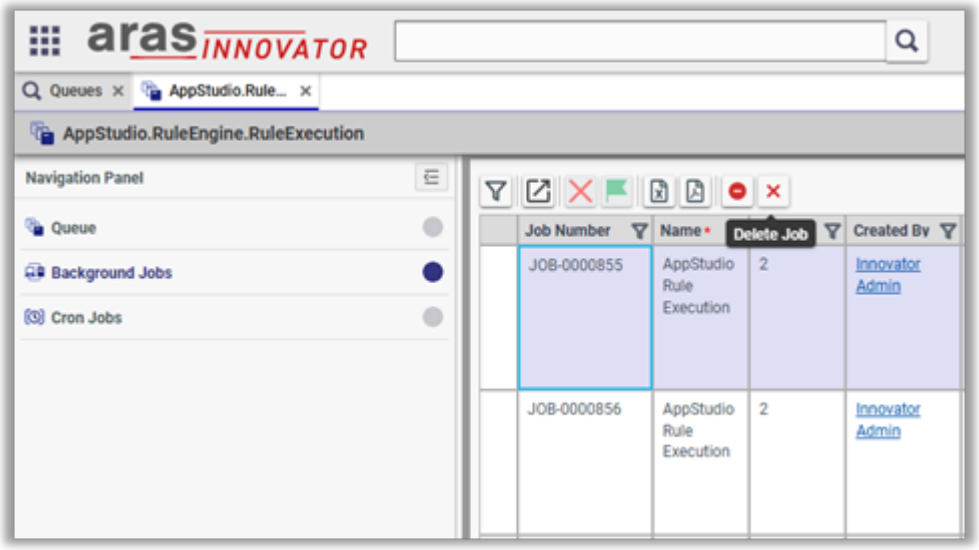
- User can open the background job to see more details



- User can "**Abort**" the execution of job only when job has Task status = "**In Progress**".
  - Right Click on Job in Queue
  - Go to Actions
  - Select "**Abort Job**".

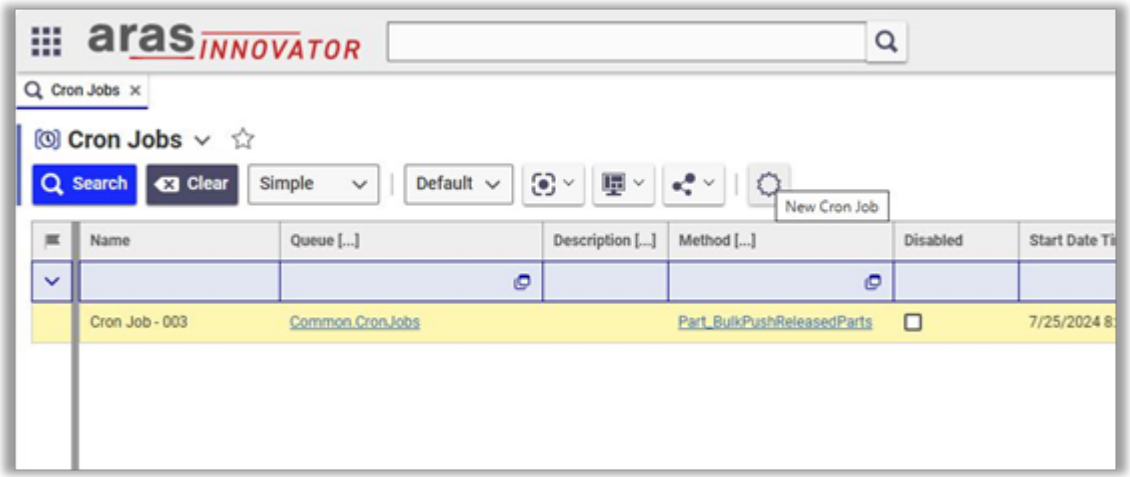


- User with Admin access can “**Delete**” the job only when job has Task status = “**Succeeded**.” or “**Discarded**” or “**Failed**”.

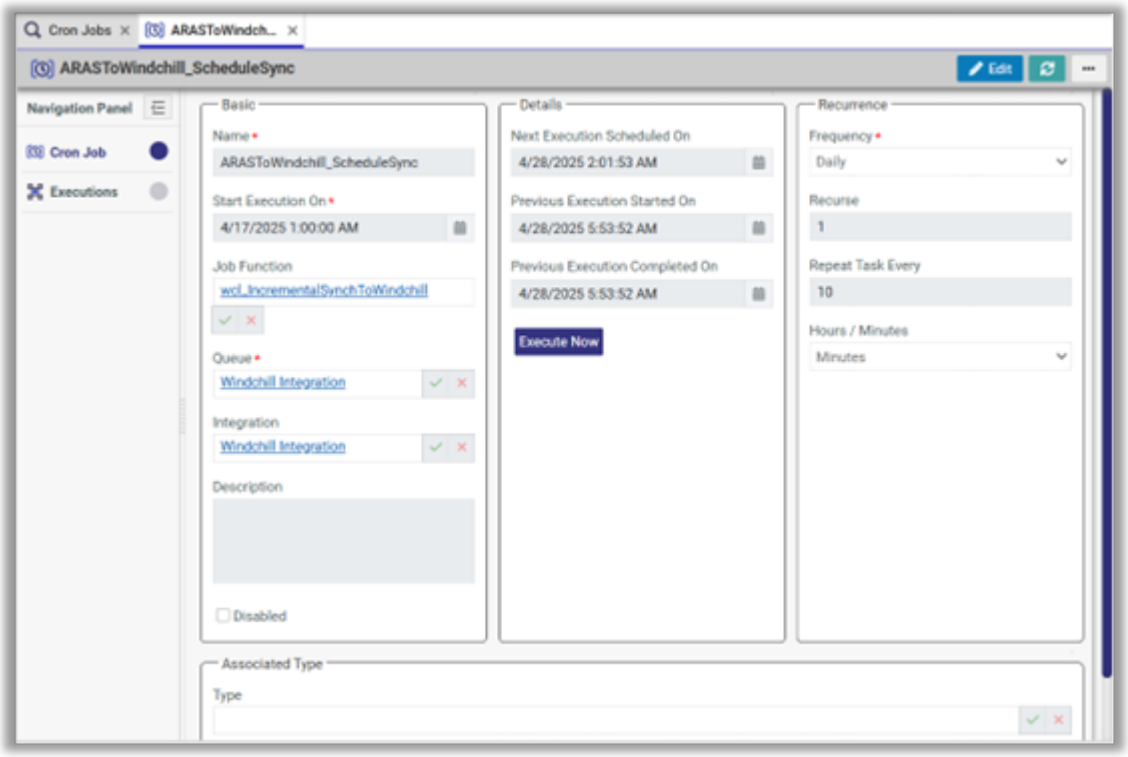


Cron Job

- Admin user can create new Cron Job going through TOC → Administrators → Pro App Designer → Background & Cron Jobs → Cron Jobs.
- Click on "**Create New Cron Job**".



- During creation, user can enter values for the Cron Job properties.

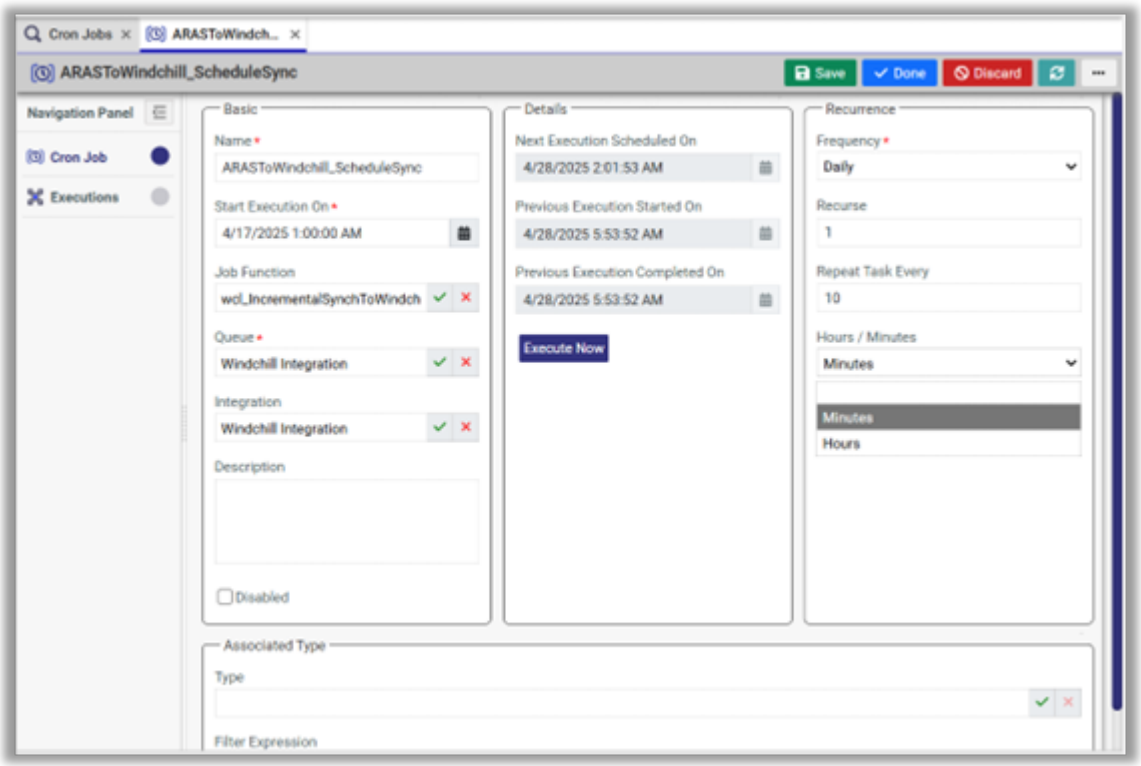


- **Name:** Name of the Cron Job
- **Start Execution On:** Initial start date time for the Cron Job

- **Job Function:** The method that you want execute as a part of Cron Job.
- **Integration (Optional)**: Integration Specific Cron Job. To be set when this Cron Job is for a specific integration.
- **Description:** Description of the Cron Job
- **Disabled:** Enable/Disable the Cron Job
- **Next Execution Scheduled On:** This is a read-only field that displays the next scheduled run time of the Cron Job, calculated automatically based on the recurrence settings configured within the job.
- **Previous Execution Started On:** This is a read-only field that displays the start time of the previous execution.
- **Previous Execution Completed On:** This is a read-only field that displays the end time of the previous execution.
- null :By clicking this button, the admin user can trigger the Cron Job to execute immediately.
- Admin user can choose the "**Frequency**" and according to the selection, relevant portions of the UI will be shown.
    - For selecting frequency of Cron Job, it has three options:
        - Daily
        - Weekly
        - Monthly

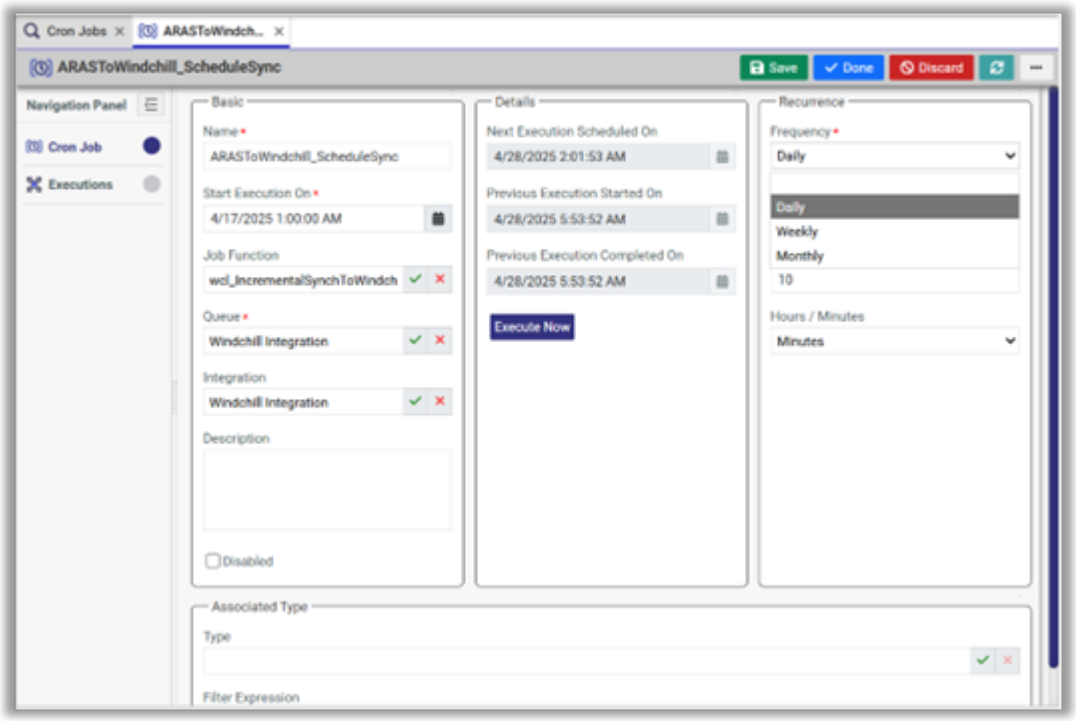
Daily:

- Recurse: It is a span of days between the Cron Job executions.
- Repeat Every Hour / minute: It is a time span for re-execution of Cron Job.



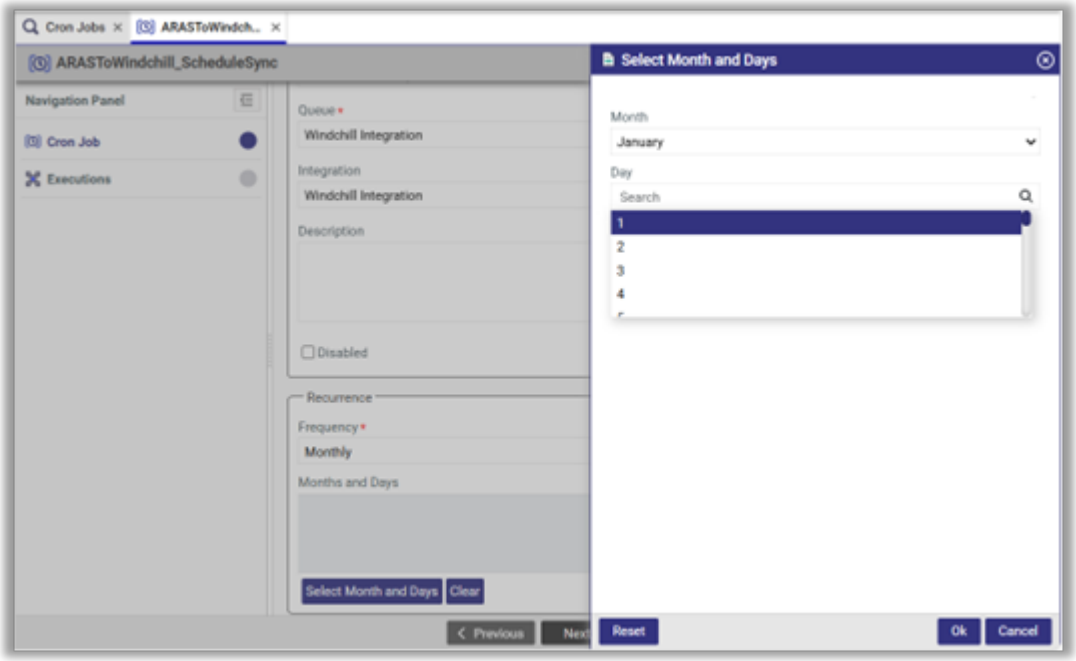
Weekly:

- Recurse: It is a span of Weeks between the Cron Job executions.
- Weekly Days: Cron Job will run on the chosen days.

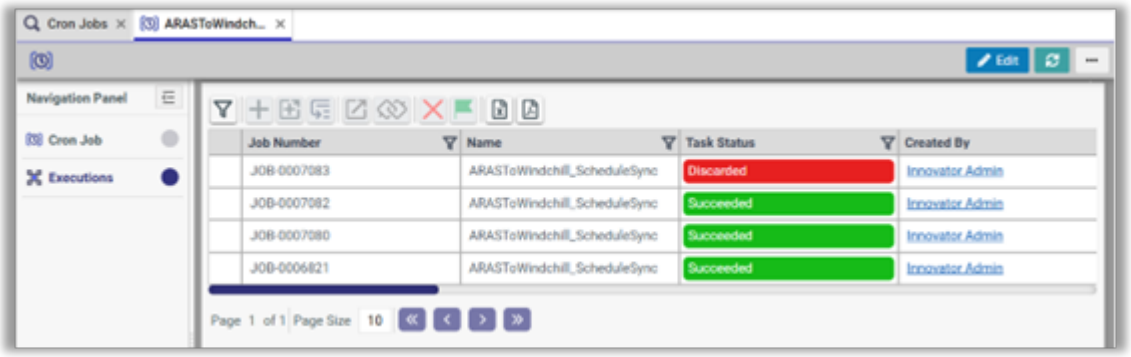


Monthly:

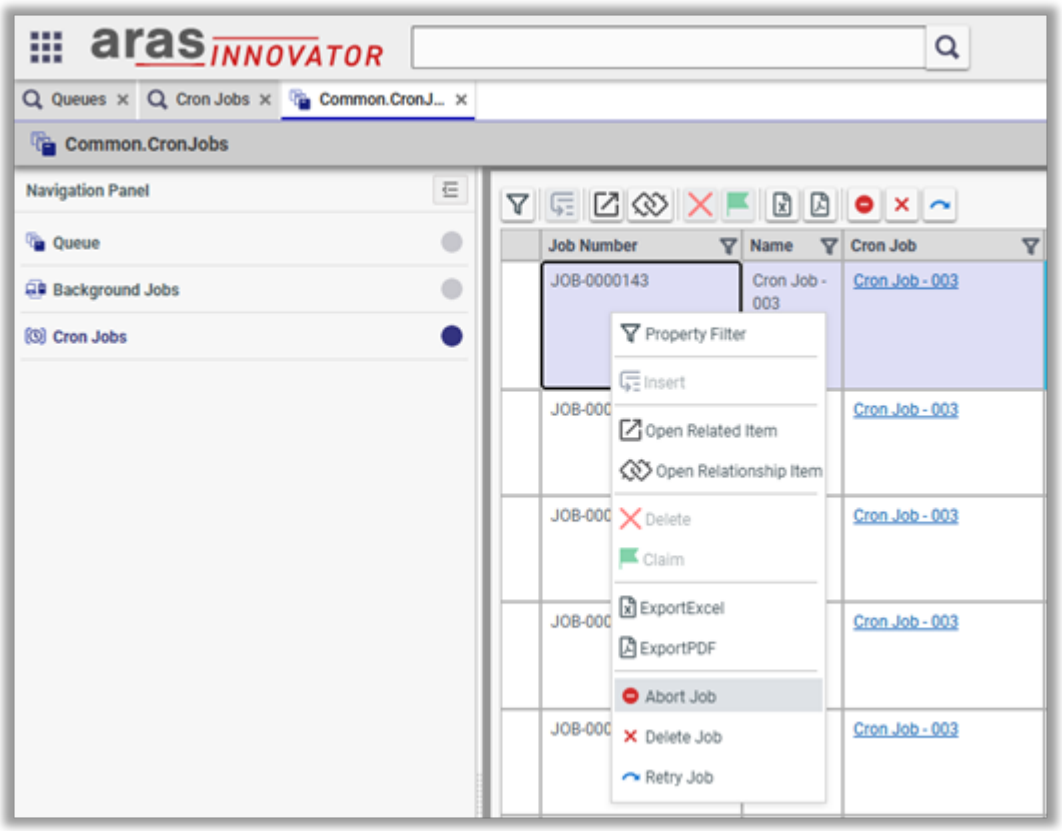
- null : By clicking on this button form "**Select Month & Day**" will pop up, After selecting month & day and clicking "**OK"** button, value will appear on the TextBox. The Cron Job will run in accordance with the chosen dates.
- null : By clicking on this button all selected Month & Day gets cleared from the TextBox.



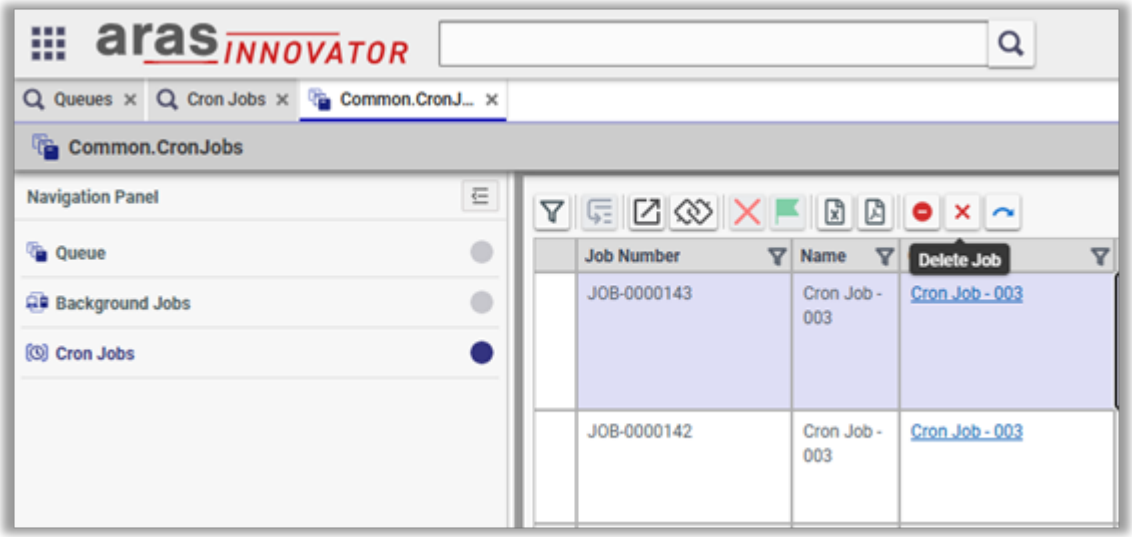
- **Associated Type → Type (Optional):** Name of the ItemType if this Cron Job is specific to any ItemType
- **Associated Type → Filter Expression (Optional):** If an Associated Type is defined for the Cron Job, this optional field allows you to specify a filter expression to be applied when retrieving items.
- **Job Executions**: User can check all the executions of a particular Cron Job by opening the Cron Job item and navigating to Executions relationships tab.



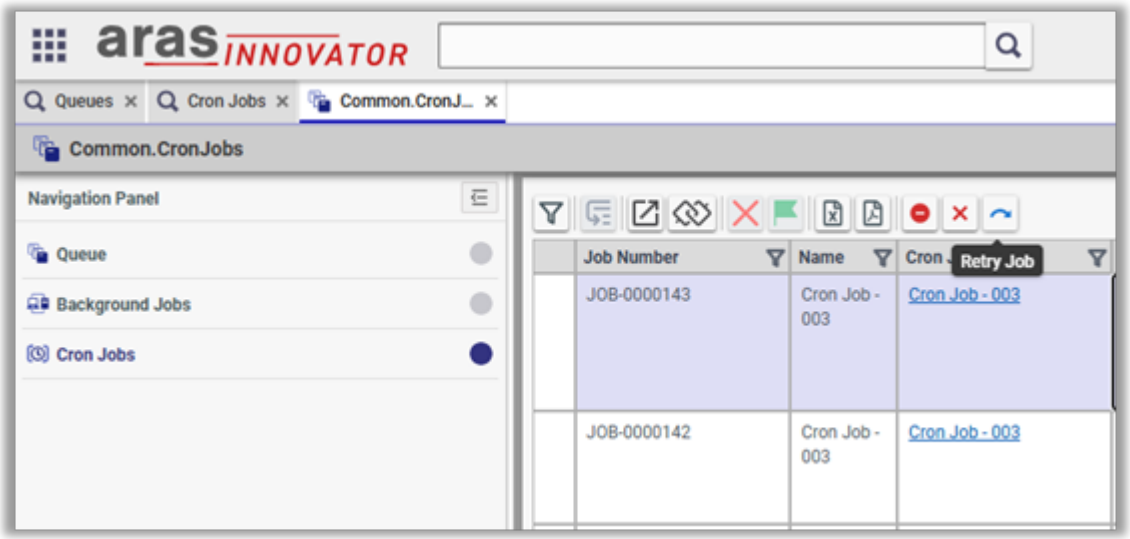
- User can “**Abort**” the execution of Cron Job only when Cron Job has Conversion Task status = “**In Progress**”.
  - Right Click on Cron Job in Queue
  - Go to Actions
  - Select “Abort Job”.



- User can “**Abort**” the execution of Cron Job only when Cron Job has Conversion Task status = “**In Progress**”.



- User can “**Retry**” the execution of Cron Job only when Cron Job has Conversion Task status = “**Failed**”.



Event Subscription and Notification User Guide

Introduction

Purpose

This guide describes how to use Aras Event Subscriptions and Notifications for external users.

Scope

This document provides instructions for configuration and functionality of Aras Event Subscriptions and Notifications.

Target Audience

This document is intended for users who will configure complex UIs and end users of these UI.

Guide Organization

This document describes how to use the Aras Event Subscriptions and Notifications. This User Guide is organized into the following sections:

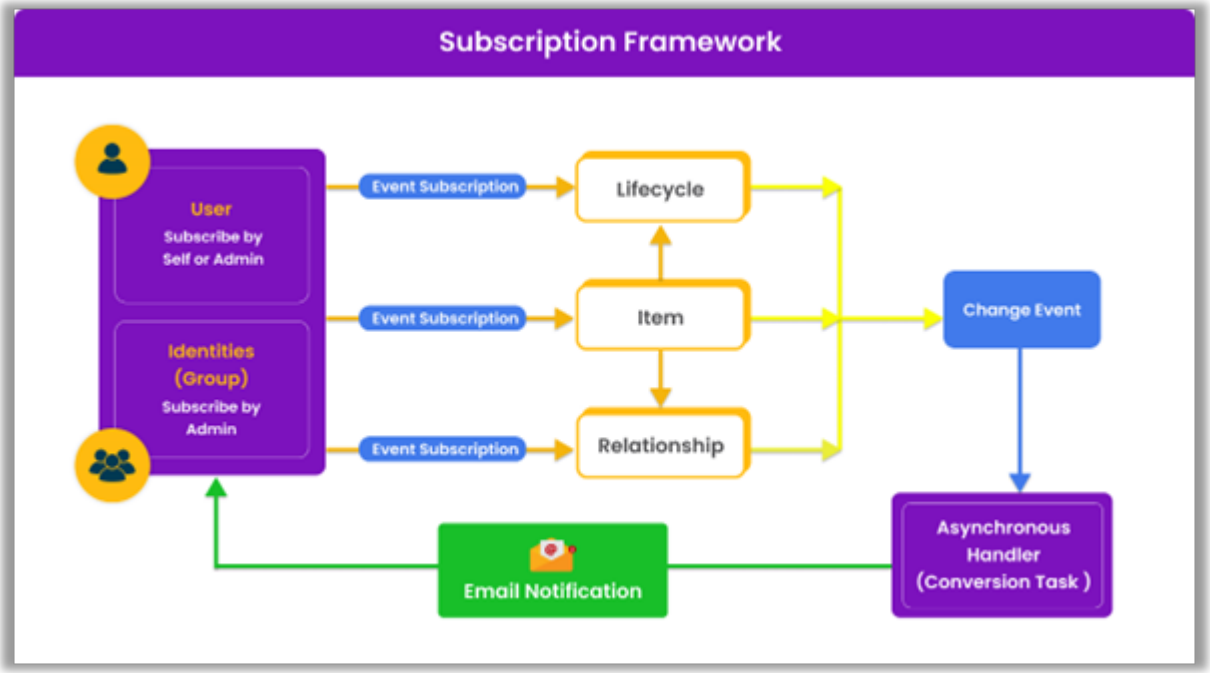
SECTION 1: **Introduction**	This section defines the purpose and scope of the User Guide, identifies its intended audience, and provides a concise summary of each part to assist users in navigating the guide effectively.
SECTION 2: **Introduction to Event Subscriptions and Notifications**	This section explains the purpose of Event Subscriptions and Notifications, offering an overview of the features available within the Aras Event Subscriptions and Notifications functionality.

Event Subscriptions & Notifications

Enables administrators or users to subscribe individuals or groups to receive notifications for various events, either at the ItemType level or the specific Item level. Additionally, users have the option to self-subscribe for notifications at the Item level.

Architecture

The Subscription Framework supports two types of subscriptions: ItemType-level and Item-Level Subscription. Administrators can subscribe multiple events to various alias or non-alias identities either at the ItemType level or for specific Items. Non-admin users also have the ability to self-subscribe to multiple events for individual Items. When a subscribed event is triggered in the system, an asynchronous handler on the conversion server sends email notifications to all subscribed users or groups. This handler operates in the background, delivering notifications to hundreds of users simultaneously without interrupting user actions. This asynchronous processing enhances the efficiency and scalability of the framework.



Event Subscriptions & Notifications Roles

When the Event Subscriptions & Notifications package is imported, it automatically deploys three default group identities: **SN Administrator**, **SN Subscription Administrators**, and **SN Employee**. The **SN Administrator** identity should be assigned to users responsible for enabling subscriptions at the ItemType level. The **SN Subscription Administrators** identity is intended for users who will manage administrative-level subscriptions, whether at the ItemType or individual Item level. Lastly, the **SN Employee** identity must be assigned to users who wish to self-subscribe to notifications for specific Items. These roles help ensure appropriate access and functionality within the subscription framework.

If all Word files have validation rules applied, the system will select the first file that successfully passes the validation. If none of the Word files meet the required validation criteria, the document generation process will be terminated.

Types Of Events Available For Subscribe

- **Admin Level Events Subscription**: Administrators have the ability to register themselves, as well as individual users or groups, to receive notifications for various events associated with a specific ItemType or Item.
- **User Level Events Subscription**: Non-admin users have the ability to register themselves to receive notifications for various events related to a specific Item.

Below are the list of events, user or admin can subscribe/unsubscribe on ItemType / Item level.

Basic Events

- **Create (Available on ItemType Level for Admin)**: When this event is subscribed to, all users configured by the administrator will receive a notification whenever an Item of the specified ItemType is created in the system.
- **Modify**: When this event is subscribed to, all users configured by the administrator at the ItemType level, along with all users who have self-subscribed at the Item level, will receive a notification whenever an Item of the specified ItemType is modified in the system.
- **Revise**: When this event is subscribed to, all users configured by the administrator at the ItemType level, as well as all users who have self-subscribed at the Item level, will receive a notification whenever an Item of the specified ItemType is revised in the system.
- **Delete**: When this event is subscribed to, all users configured by the administrator at the ItemType level, along with all users who have self-subscribed at the Item level, will receive a notification whenever an Item of the specified ItemType is deleted from the system.

Relationship Events

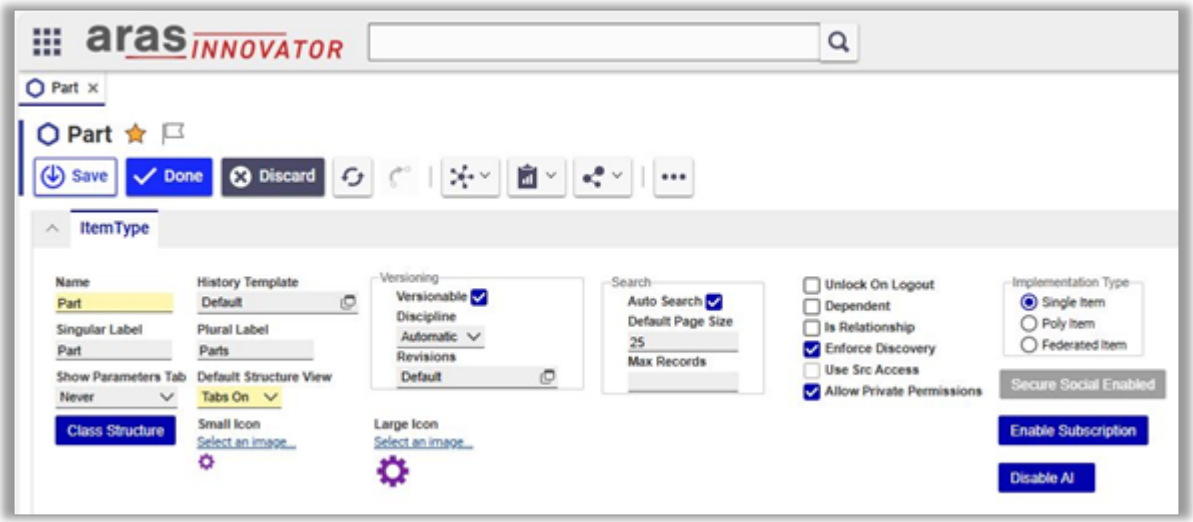
- On subscription window, events are shown to the user/admin dynamically inside *Relationship* section.
- This section displays a list of non-TGV relationships associated with the context ItemType. It allows users and administrators to subscribe to one or more of these relationships, either at the ItemType level or the individual Item level.
- For all subscribed relationships, notifications will be sent to users configured by the administrator at the ItemType level, as well as to users who have self-subscribed at the Item level, under the following conditions.
    - When new relationship is created in the system.
    - When any relationship is deleted from the system.

Lifecycle Events

- Within the subscription window, available events are dynamically displayed to the user or administrator inside *Lifecycle States* section.
- This section displays the list of lifecycle states associated with the context ItemType. It enables both users and administrators to subscribe to one or more of these lifecycle states, either at the ItemType level or the individual Item level.
    - For all subscribed lifecycle states, notifications will be sent to users configured by the administrator at the ItemType level, as well as to users who have self-subscribed at the Item level, whenever an Item enters a subscribed state.

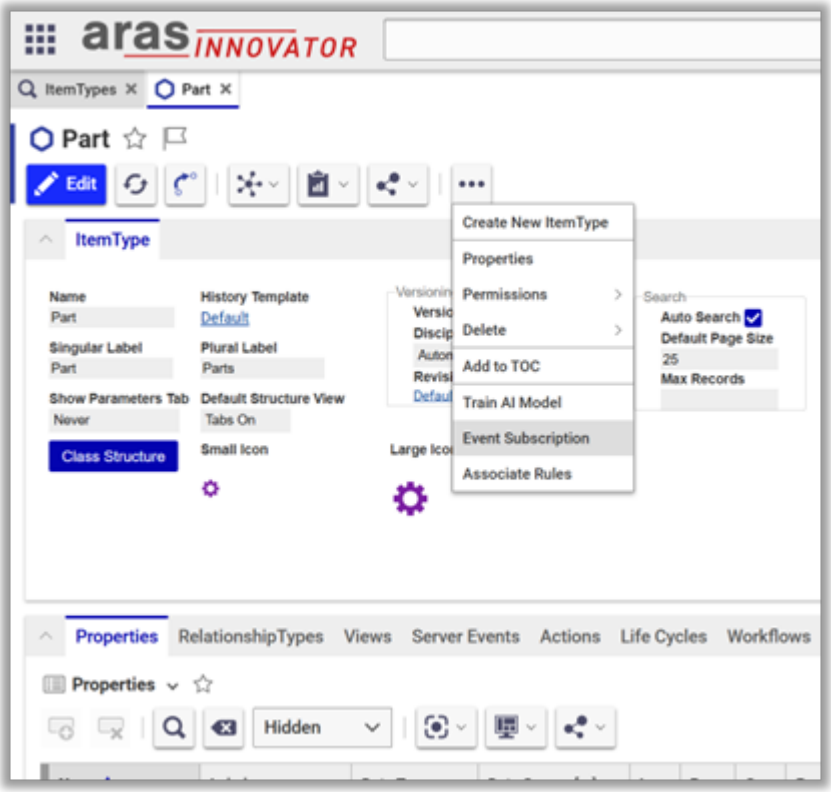
Enable Subscription on an ItemType

Admin user can enable the subscription on an ItemType by just opening the ItemType from TOC -> Administration -> ItemTypes -> Search and open the ItemType -> Click on null

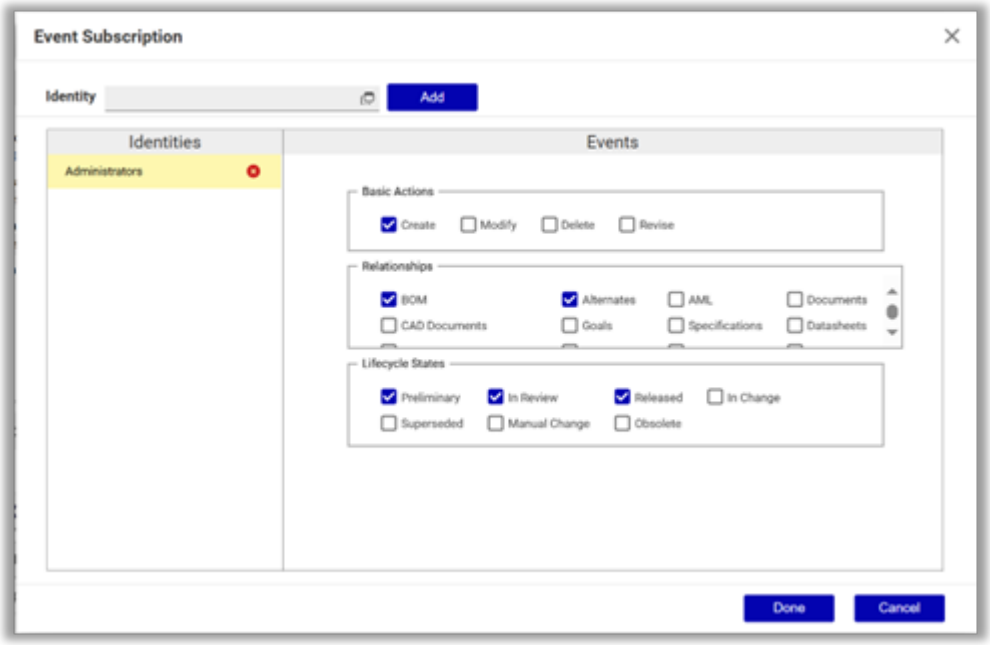


Configure Subscriptions on an ItemType

Once subscription is enabled on an ItemType, the administrator can utilize the **Event Subscription** action to configure admin-level subscriptions.



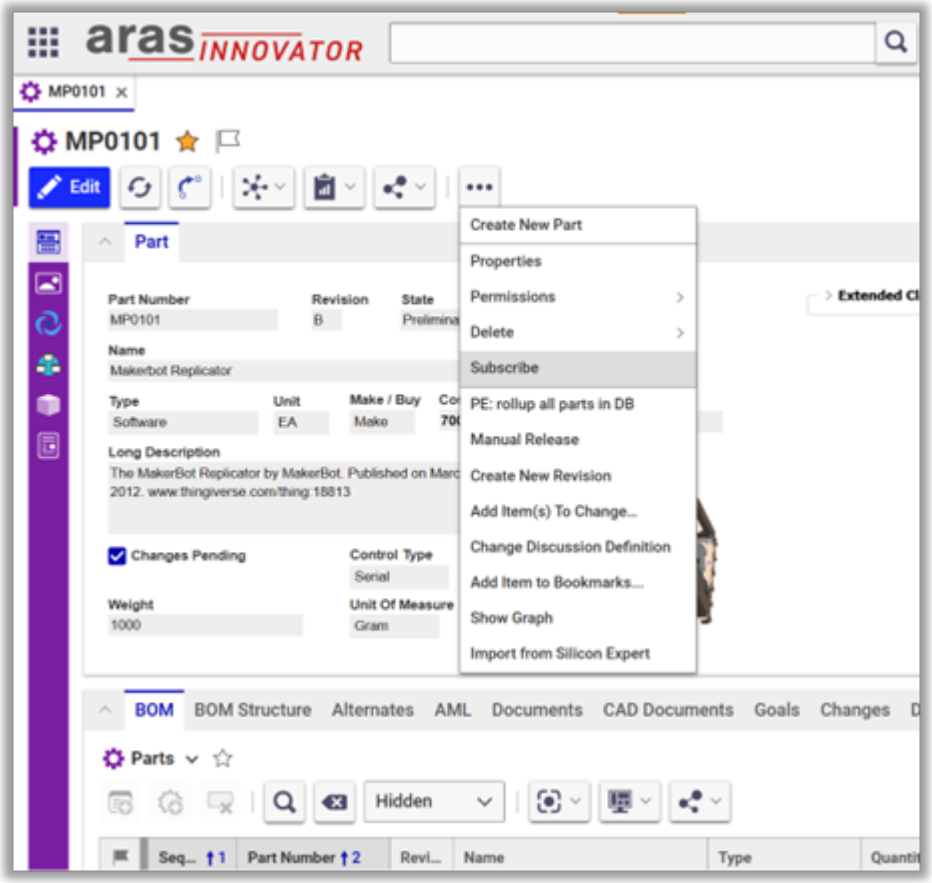
- This will open the below dialog box.



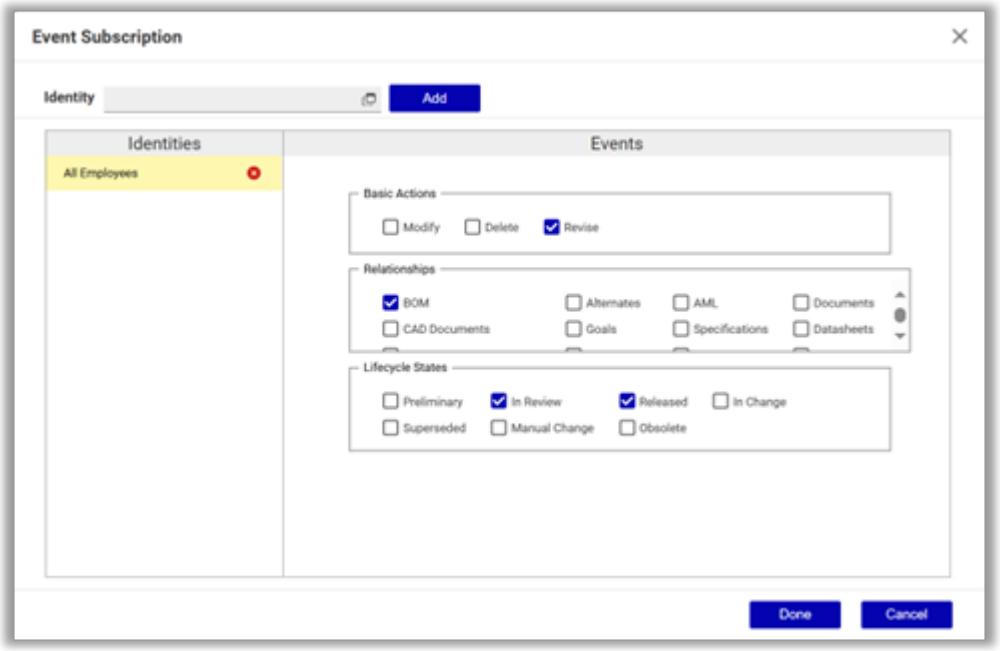
- When this dialog box is loaded, it automatically populates the list of identities along with their corresponding subscriptions.
- Admin user can add the new Identity (Alias or Non-alias) using the **Add** button.
- Admin user can select the Identity on left panel and add the events subscription on right panel.
- Admin user can update the event selections for the selected Identity.
- Admin user can remove/delete the selected Identity from admin subscriptions.
- Clicking on **Done** button saves the final state of event subscriptions for all identities into the system and closes the dialog box.
- Clicking on **Cancel** button will abort the operation and close the dialog box.

Configure Subscriptions on an Item

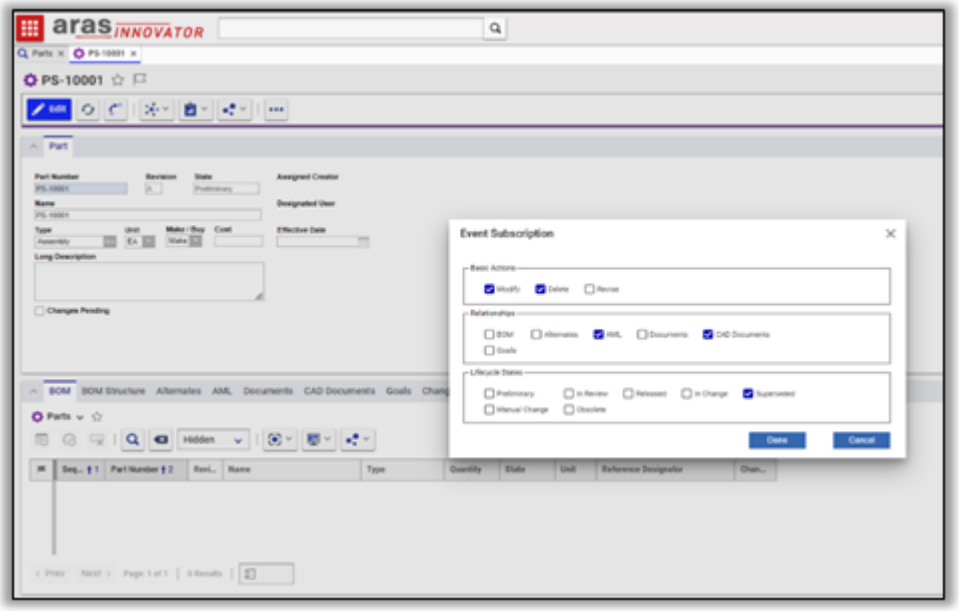
- Open any subscription enabled ItemType's Item using subscription admin user and click on **Subscribe** button.



- For admin subscription users, it will open below dialog box where admin user can add the admin level subscriptions on Item.

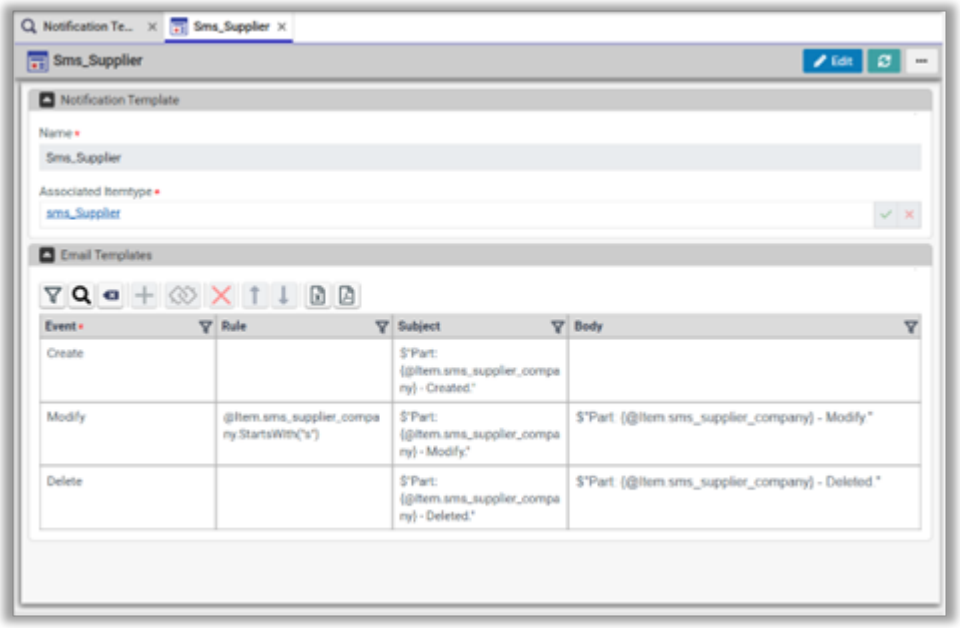


- For non-admin users, it will open below dialog box where user can add the subscriptions for themselves on an Item.



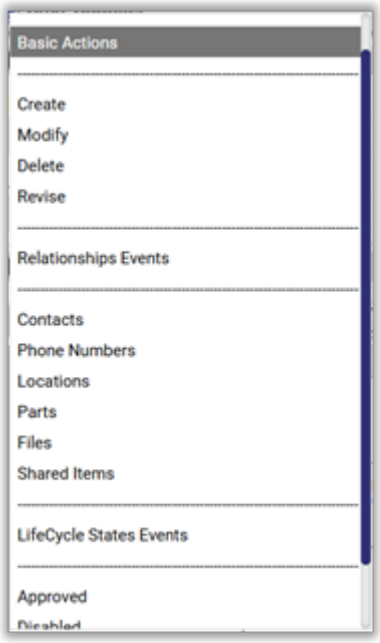
Notification Templates

- Notification Templates are used to define the content of email notification messages. These templates allow you to configure both the subject and the body of the message for different Events mentioned in section Types of events available for subscribe .
- Admin users can setup Notification Templates for each ItemType.
- To configure Notification Templates, go to TOC → Administrators → ProAppDesigner → Templates → Notification Templates.



- Below are the properties that needs to be set:
  - **Name:** Name of the template
  - **Associated ItemType:** ItemType for which this template is configured.
- In the Email Templates section below, the administrator can configure the email message for each event. During this setup, the user is required to specify the following key properties.

- **Event:** You can select any of the Event listed in the dropdown.



- **Rule:** Admin can configure Rules based on Item's and User's properties.
- **Subject:** The Subject field defines the title of the email message. Administrators can customize it by including properties from the related Item using the following syntax:

```
$"Part: {@Item.sms_supplier_company} - Created."
```

- **Body:** The Body of the email message contains the main content delivered to recipients. Administrators can customize this section by including properties from the related Item using the syntax:

```
$"Part: {@Item.sms_supplier_company} - Modify."
```

ArasProApp Designer - SAAS Installation Guide

Introduction

Purpose and Target Audience

The **Aras ProAppDesigner SaaS Installation Guide** for Aras DevOps provides instructions for Aras DevOps and Enterprise subscribers to add the Aras ProAppDesigner application to their Aras Innovator using SaaS. This document is intended for Aras DevOps users responsible for installing and configuring the applications and platform components for the Aras Innovator instance utilizing SaaS.

Prerequisites

To install Aras ProAppDesigner, **users must confirm** that the installation of Aras Innovator meets the minimum requirements to run ProAppDesigner.

Check the eligibility of code tree and database:

From the **TOC**, select **Administration**, **Variables**.

VersionMajor	VersionMinor	VersionServiceUpdate
14	0	22
14	0	23
14	0	24
14	0	25
14	0	26

If it does not meet the service pack requirements listed in the previous steps, contact Aras to discuss options.

Aras DevOps administrators or contributors **must have**:

- An installed Aras Innovator instance before following the steps outlined in this document.
- It is **recommended** that all administrators read the ***Aras DevOps – User Guide***. It explains how to utilize the Aras DevOps service effectively.

Definitions

This section defines the terms used in this document.

Term	Definition
Code tree	The set of files that make up the application files for Aras Innovator.
Code tree patch	The set of files in an application installation package applied to the code tree to install the code tree component.
Config transformation	The transformation sourced into the project repository corresponding to changes in configuration files of Aras Innovator code tree.
Import package	The set of XML files and the corresponding import.mf file representing a package or set of packages that need to be installed to apply the database changes of an application.
Installation package	The set of files utilized to install an application (may include a code tree patch, an import package, and/or converters).
Platform Component	Not release-specific components that can be installed without a direct dependency on the installed Aras Innovator release.

Aras ProAppDesigner Installation

Aras Innovator needs to be installed before **Aras ProAppDesigner** component. It is **important** to notify users that Aras Innovator will experience downtime due to installation. Additionally, it is important to **create backups** for the Aras Innovator code tree and database.

Important

Please store backup files securely, to restore the system to its pre-installation state if installation fails.

Aras ProAppDesigner Installation Instruction

The following steps outline the process of installing **Aras ProAppDesigner**:

1. Download the **Aras ProAppDesigner CD Image** from the Aras File Sharing site.
2. Unzip the Aras ProAppDesigner CD Image on the local machine.
3. Navigate to the `\Files\` folder of the Aras ProAppDesigner CD Image.
4. Copy the contents of the Innovator folder into the \CodeTree\Innovator\ folder of the `Work.git` repository.
5. Copy the contents of the `ConversionServer` folder into the `\CodeTree\ConversionServer\folder` of the `Work.git` repository.
6. If any of the below DLLs are available in `\Innovator\Server\bin or in ConversionServer\bin` folder, delete them.

- Prorigo.Protrak.ExpressionEvaluatorStd.Impl.dll
- Prorigo.Protrak.ExpressionEvaluatorStd.dll
- Prorigo.Plm.PresentationGenerator.dll
- Prorigo.Plm.PresentationGenerator.Aras.dll
- Prorigo.Plm.Logging.Framework.dll
- Prorigo.Plm.DocumentGenerator.dll
- Prorigo.Plm.DocumentGenerator.Aras.dll
- Prorigo.Plm.DiscussionPanel.dll
- Prorigo.Plm.Core.dll
- Prorigo.Plm.Aras.BackgroudCronJobFramework.dll
- Prorigo.Plm.AppStudio.RuleEngine.Impl.dll
- Prorigo.Plm.AppStudio.RuleEngine.dll
- Prorigo.Plm.AppStudio.IntegrationFramework.dll
- Prorigo.Plm.AppStudio.dll

7. Copy the Pro App Designer folder, except the `imports.mf` file, from the `..\Imports\Pro App Designer..` folder into the `AML-packages` folder of the `Work.git` repository.

> **Important**
>
> Do not copy the imports.mf file, as it will overwrite the existing imports.mf file in the Work.git repository.

8. Open the `..\Imports\imports.mf` file.
9. Copy the following <package> elements:

```
<imports>
  <package name="appstudio" path="ProAppDesigner\appstudio\Import" />
  <package name="documentgenerator" path="ProAppDesigner\documentgenerator\Import" />
  <package name="technicaldatapackage" path="ProAppDesigner\technicaldatapackage\Import" />
  <package name="businessruleengine" path="ProAppDesigner\businessruleengine\Import">
    <dependson name="backgroundcronjobframework" />
  </package>
  <package name="backgroundcronjobframework" path="ProAppDesigner\backgroundcronjobframework\Import"
  <package name="subscriptionframework" path="ProAppDesigner\subscriptionframework\Import" />
  <package name="powerpointgenerator" path="ProAppDesigner\powerpointgenerator\Import" />
  <package name="appstudiotemplates" path="ProAppDesigner\appstudiotemplates\Import">
    <dependson name="appstudio" />
  </package>
  <package name="integrationframework" path="ProAppDesigner\integrationframework\Import">
    <dependson name="backgroundcronjobframework" />
  </package>
  <package name="discussionpanel" path="ProAppDesigner\discussionpanel\Import" />
  <package name="proappdesigner_toc" path="ProAppDesigner\proappdesigner_toc\Import">
    <dependson name="appstudio" />
    <dependson name="backgroundcronjobframework" />
    <dependson name="businessruleengine" />
    <dependson name="documentgenerator" />
```

```
      <dependson name="integrationframework" />

      <dependson name="powerpointgenerator" />

      <dependson name="subscriptionframework" />

      <dependson name="technicaldatapackage" />

    </package>

  </imports>
```

10. Paste the `<package>` elements into the `imports.mf` file of the Work.git repository.
11. Copy `PreImportAMLScripts\04500_TemplateDeploy_PreAML.xml` to the `AmlDeploymentScripts\1-BeforeAmlPackagesImport` directory.
12. Copy `PostImportAMLScripts\05000_TemplateDeploy_PostAML.xml` to the `AmlDeploymentScripts\2-AfterAmlPackagesImport` directory.
13. Modify `method-config.xml` available under `TransformationsOfConfigFiles\Innovator\Server`
14. Add below `ReferencedAssemblies` entries

```
  <MethodConfig xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
  <ReferencedAssemblies>
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
```

```
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.A
  <name
xdt:Transform="InsertAfter(/MethodConfig/ReferencedAssemblies/name[last()])">$(binpath)/Prorigo.Plm.L
  </ReferencedAssemblies>
  </MethodConfig>
```

Important

If method.config file does not exist at above location, add that file and put entire content given here.

Important

If any of the DLLs mentioned in the DLLs to be deleted section, please delete the corresponding entry from method-config.xml file.

15. Stage and review the changes to ensure that no customizations are overwritten.
16. Commit the changes.
17. Run the **continuous-integration** Pipeline.
18. Run the **deploy-innovator** Pipeline to complete the installation.

Confirming the Installation

Use the following procedure to check if the database has been updated correctly:

1. **Log in** to Aras Innovator as an administrator.
2. From the **Table of Contents**, expand **Administration** and select **Variables**.
3. Confirm the following variable:

Pro App Designer
14.14.22

If the installation fails at any time, **revert** to backups and **contact Aras Support** at support@aras.com.

ArasProApp Designer - Upgrade Guide

Overview

This document outlines the steps for upgrading Aras ProAppDesigner from any previous release to 14.14.*. These changes affect both the server database and code tree.

Prerequisites

To upgrade from Aras ProAppDesigner 14.14.* confirm eligibility for the upgrade. Download and install any required software.

Checking for Eligibility

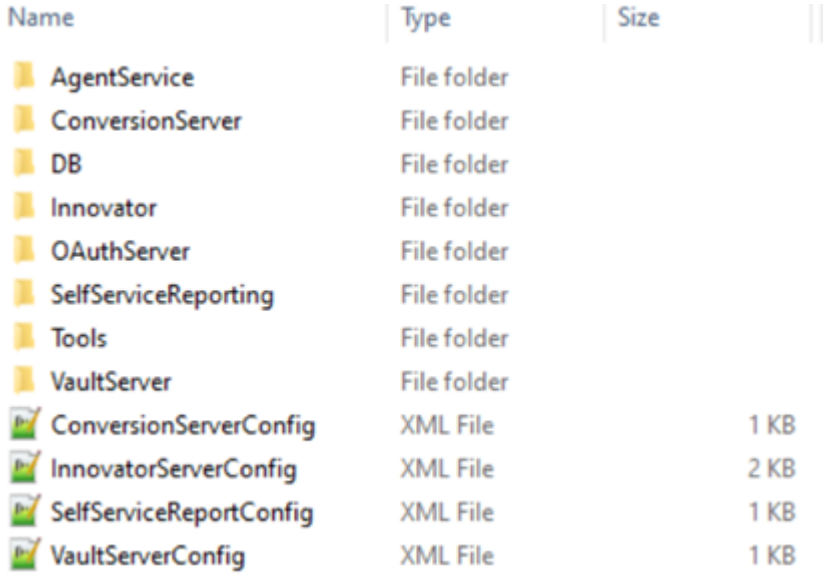
Please use the following procedure to check that your Aras Innovator code tree and database meet the minimum requirements to apply the Aras ProAppDesigner Solution AML.

1. Log in to Aras Innovator as an administrator.
2. From the TOC, select **Administration\Variables**.
3. Confirm following variable value:
   Pro App Designer: 14.0.1

Upgrading Aras ProAppDesigner

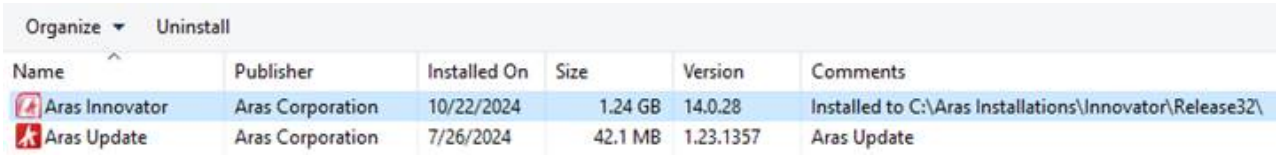
Notification and Backup

1. Notify users that the system will be down at a scheduled time and that they should log out before the process starts.
2. Backup the Code Tree.
   The “Code Tree” refers to files and folders installed on the disk when Aras Innovator was first installed.
   The default path for the Code Tree installation would be something like:
   C:\Program Files (x86)\Aras\Innovator
   With the following contents:



   The following steps can be used to confirm the installation folder:

   1. From the Windows search field, enter **Control** to access the **Control Panel**.
   2. Select **Programs and Features**.
   3. Right-click the **Name** column and select **More…**
      - Select the **Comments** header if it has not been checked.
      - Click **OK**
   4. Search for the **Aras Innovator** entry in the **Programs and Features** window.
   5. View the value set in the **Comments** column for the Aras Innovator entry:



3. Disconnect all users from the database.
   The easiest way to prevent client sessions from committing any further changes to the database is to change the database connection string in the InnovatorServerConfig.xml from <DB-Connection... to <xDB-Connection… and restart the **w3svc**

service (IIS). This expires all sessions and prevents all new connections to the Aras Innovator database through the existing instance.

4. Backup the database.
   Store these files in a safe location, as they will need to be restored if the process fails.

5. Enable database connections.
   After the database backup has been completed, enable the database connection string in the `InnovatorServerConfig.xml` by changing it from `<xDB-Connection...` to `<DB-Connection...` and restarting the **w3svc service (IIS)**.

Upgrading Aras ProAppDesigner

1. Download the Aras ProAppDesigner CD Image from the Aras FTP site, in the `Aras ProAppDesigner Releases` folder, and unzip the file on the local computer.
2. From the root of your install directory, navigate to `Innovator\Client\scripts\AppStudio` folder and delete all its content
3. From the root of your install directory, navigate to Innovator\Server\bin folder and delete below listed DLLs
    1. Prorigo.Protrak.ExpressionEvaluatorStd.Impl.dll
    2. Prorigo.Protrak.ExpressionEvaluatorStd.dll
    3. Prorigo.Plm.PresentationGenerator.dll
    4. Prorigo.Plm.PresentationGenerator.Aras.dll
    5. Prorigo.Plm.Logging.Framework.dll
    6. Prorigo.Plm.DocumentGenerator.dll
    7. Prorigo.Plm.DocumentGenerator.Aras.dll
    8. Prorigo.Plm.DiscussionPanel.dll
    9. Prorigo.Plm.Core.dll
    10. Prorigo.Plm.Aras.BackgroudCronJobFramework.dll
    11. Prorigo.Plm.AppStudio.RuleEngine.Impl.dll
    12. Prorigo.Plm.AppStudio.RuleEngine.dll
    13. Prorigo.Plm.AppStudio.IntegrationFramework.dll
    14. Prorigo.Plm.AppStudio.dll
4. From the root of your install directory, navigate to ConversionServer\bin folder and delete below listed DLLs
    1. Prorigo.Plm.Logging.Framework.dll
    2. Prorigo.Plm.Core.dll
    3. Prorigo.Plm.Aras.BackgroudCronJobFramework.dll
    4. Prorigo.Plm.AppStudio.IntegrationFramework.dll
5. Copy the `\Files\Innovator\Client` folder to the root of your installation directory. When prompted, **overwrite** the existing `\Innovator\Client` folder and all of its contents.
6. Copy the `\Files\ConversionServer` folder to the root of your install directory, overwriting the existing `\ConversionServer` folder and all its contents.
7. From the root of your code tree, navigate to `\Innovator\Server` and edit the `method-config.xml` file.
    1. Add below entries as shown:

       ...

```
<!-- ProAppDesigner DLLs : Start -->
<name>$(binpath)/Prorigo.Plm.Aras.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.AppStudio.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.TDP.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.SubscriptionFramework.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.DocumentGenerator.dll</name>
```

```
<name>$(binpath)/Prorigo.Plm.Aras.BackgroundCronJobFramework.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.RuleEngine.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.PresentationGenerator.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.IntegrationFramework.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.IntegrationFramework.Connector.dll</n
<name>$(binpath)/Prorigo.Plm.Aras.DiscussionPanel.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.Logger.dll</name>
<name>$(binpath)/Prorigo.Plm.LicensingFramework.dll</name>
<!-- ProAppDesigner DLLs : End -->
</ReferencedAssemblies>
```

2. Save the changes to the method-config.xml file and close it

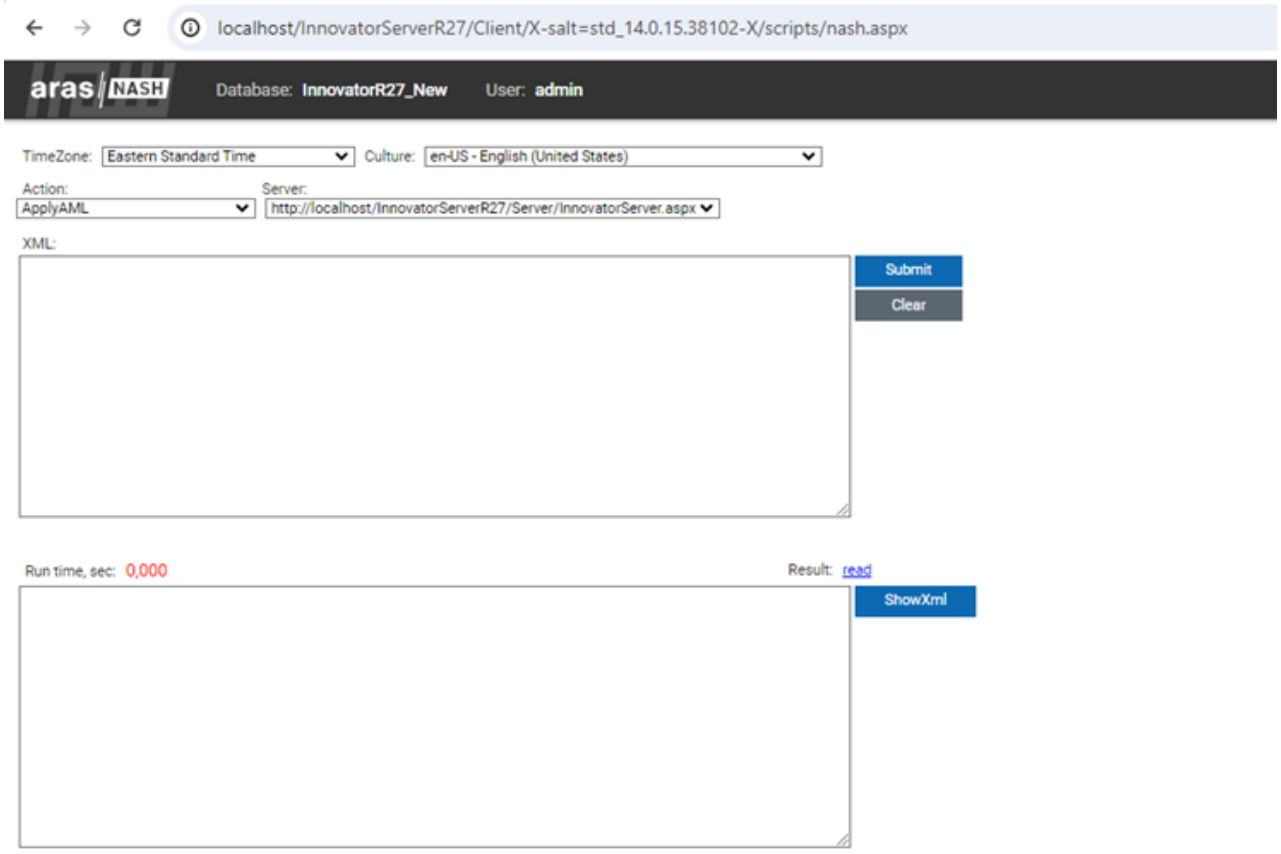
> **Important**
>
> If any of the DLLs mentioned in the DLLs to be deleted section, please delete the corresponding entry from method-config.xml file.

8. Execute **Pre-Import Scripts**

   Before you install Aras ProAppDesigner, execute below scripts:

   1. Navigate Aras Nash tool using URL http://localhost/InnovatorServer/Client/**scripts/nash.aspx**

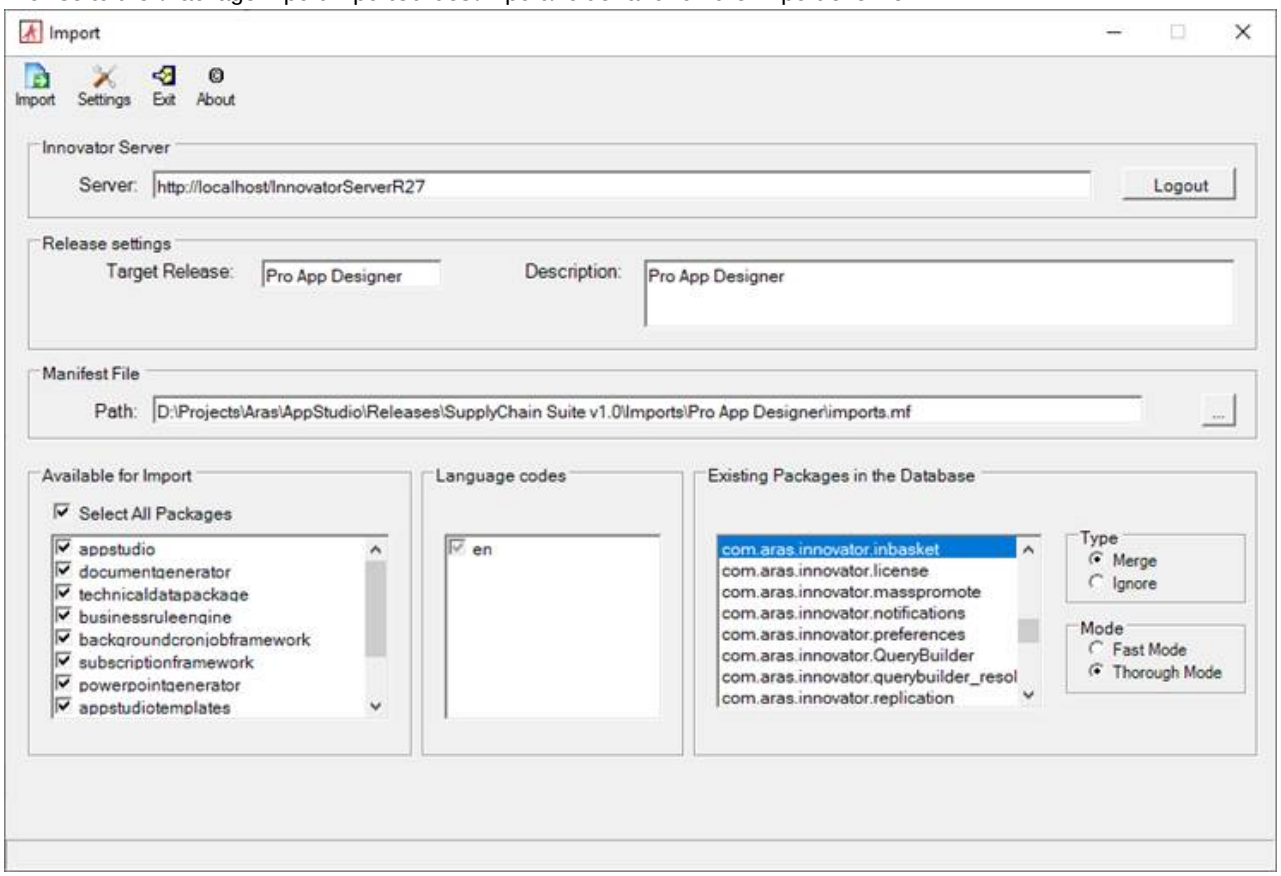
2. Login in Nash using **admin** user.



3. In the package go to folder PreImportAMLScripts, open each file in below sequence, copy the content in XML form and click on Submit.
    - 04500_TemplateDeploy_PreAML.xml

9. Complete the **Aras ProAppDesigner** Installation: Import the **ProAppDesigner** database package with the Package Import Export Utilities (For more information using this tool, please see the *Aras Innovator - Package Import Export Utilities* documentation.)
    1. You will need to enable the “Super User” login to perform this step.

2. Browse to the \PackageImportExportUtilities\Import\ folder and run the Import.exe file.



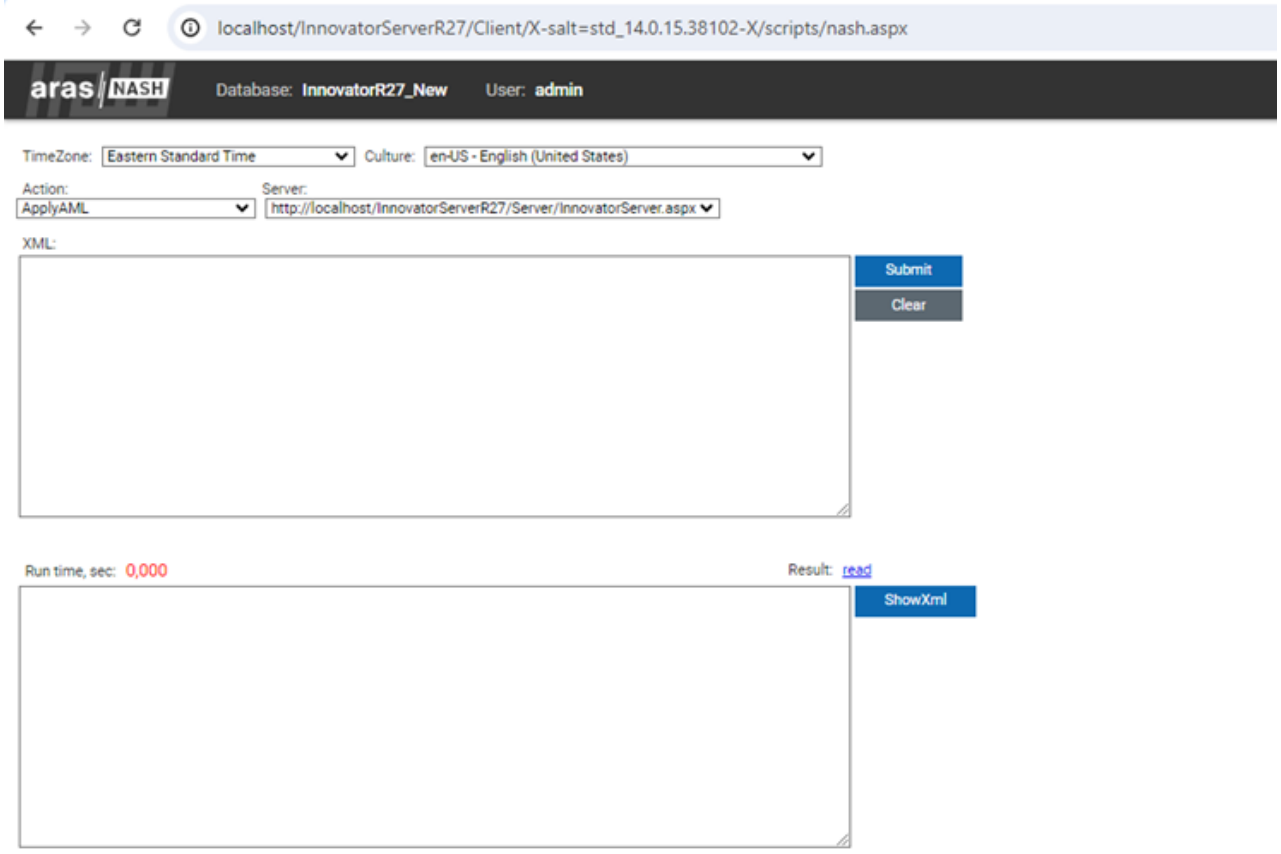
3. Enter the following information:
   - **Server** = The connection URL for Aras Innovator
   - **Database** = The target Aras Innovator database (login screen)
   - **Username** = root (login screen)
   - **Password** = Password for “root” login (Default is “innovator”) (login screen)
   - **Target Release** = ProAppDesigner
   - **Description** = ProAppDesigner
   - **Manifest File Path** = The manifest is found in the unzipped patch folder in \Aras ProAppDesigner CD Image\Imports\ProAppDesigner\imports.mf
   - **Available for Import** = Check all packages
   - **Type** = Merge
   - **Mode** = Thorough Mode
4. Click the Import button.

10. Disable the “Super User” login.
11. Execute **Post Import Scripts**

    After you install Aras ProAppDesigner, execute below scripts:

    1. ̲Navigate Aras Nash tool using URL http://localhost/InnovatorServer/Client/**scripts/nash.aspx**

2. Login in Nash using **admin** user.



3. In the package go to folder PostImportAMLScripts, open each file in below sequence, copy the content in XML form and click on Submit.
    - 05000_TemplateDeploy_PostAML.xml

Confirming the Installation

Use the following procedure to check if your database has been updated correctly:

1. Log in to Aras Innovator as an administrator.
2. From the TOC, select **Administration à Variables**.
3. Confirm the following sets of variables are listed:

Pro App Designer
14.14.*

Important

* here represents version of Innovator.

If at any time the installation fails, revert to your backups and contact Aras Support at support@aras.com.

ArasProApp Designer - User Guide

Introduction

Purpose and Target Audience

This User Guide describes how to use the Prophesier for external users with limited access data.

Content Overview

This document describes how to use Aras ProAppDesigner. This User Guide is organized into the following sections:

SECTION 1: Introduction	This section outlines the purpose and scope of this User Guide, as well as its intended target audience. It supplies a brief overview of each section to help users in navigating the guide.
SECTION 2: Introduction to Aras ProAppDesigner	This section outlines Aras ProAppDesigner information.
SECTION 3: Global Dashboard	This section provides information about global dashboard
SECTION 4: Type Dashboard	This section provides information about type specific dashboard
SECTION 5: Wizard	This section describes wizard actions and pages.
SECTION 6: Page Controls	This section provides detailed information about page controls.
SECTION 7: Page & Control Features	This section provides detailed information about page controls.
SECTION 8:	This section provides information about data export from Aras ProAppDesigner.

Data Export **SECTION 9:** **Responsive UI and Mobile Views**	This section provides information about responsive UI and mobile views features of Aras ProAppDesigner.

Introduction to Aras ProAppDesigner

Aras ProAppDesigner Overview

It is a **designer** with a rich set of **UI controls and backend components** to build applications in Aras Innovator just by configuration. Its benefits are:

- It is a **no-code solution** to generate complex, responsive, mobile friendly UI.
- It has a rich set of controls for building complex and modern UI. Any multi-page application can be **built rapidly** in studio environment with drag-drop capabilities.
- Organizes complex UI in to **structured containers** like **Pages**, **Dialogs**, **Sections**, **Groups** which intern enable UI to adapt any screen size.
- It enables creating **guided UI** for showing long data entry forms with conditional display on UI elements based on the configuration.
- It has **rich features** like **conditional display**, **validation rules**, **default values**, **conditional formatting** all with configuration. You don’t have to be a programmer to build application with Aras ProAppDesigner.

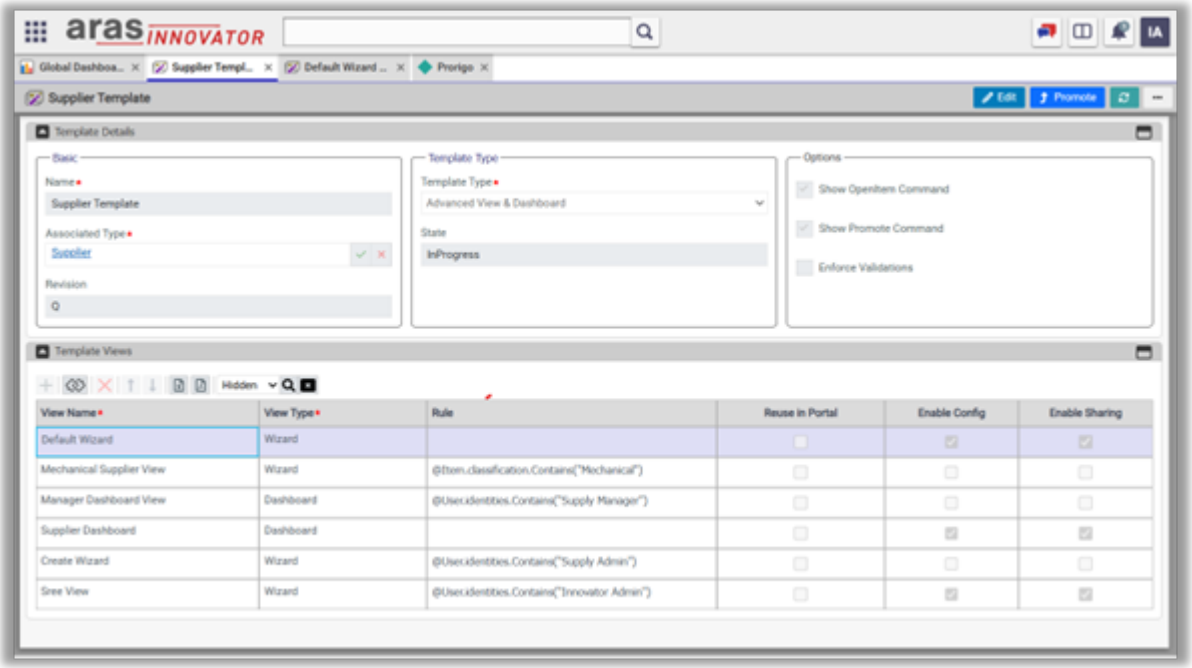
Purpose of Aras ProAppDesigner Wizard

Aras ProAppDesigner Wizard is a rich user interface used to show details of the item at runtime.

Process Overview

To render any item using Aras ProAppDesigner wizard one needs to follow the steps below:

- **Create Template** or **View for the ItemTypes** that need to be rendered using wizard.
- **Publish** the template through Lifecycle.
- **Create**, **view** and **edit** actions from Innovator now use published Template for creating or accessing the item.



Global Dashboard

Global dashboard shows data related to multiple ItemTypes in the form of table, worksheet, and report controls. You can have a table or worksheet show list of items from a specific ItemType with pre-applied filter. You can also have reports generated based on different ItemTypes.

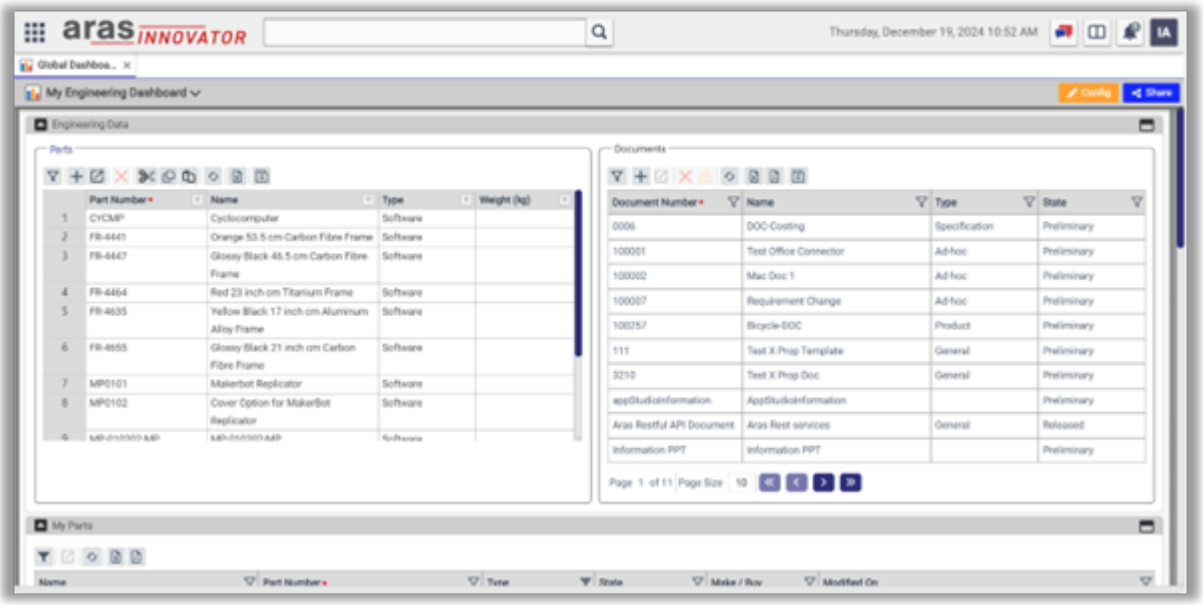


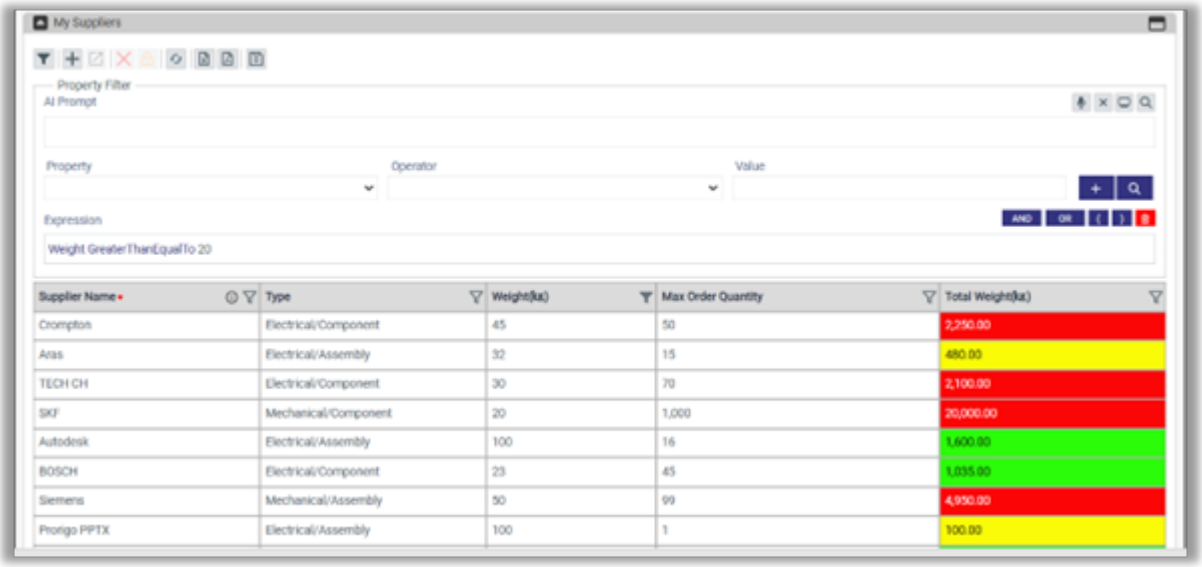
Table Control

Table control can render items of an ItemType based on the pre-applied filter. You can modify the filter if admin configures the table with "Enforce Filter" as false. It has features like:

- **Computed Columns**
- **Validation Rules**
- **Conditional Formatting**
- **Variable row height**; row height will be adjusted based on the cell content
- **Cell navigation using navigation keys** and all other shortcut keys are allowed for operations like Copy, Paste, Delete etc.
- **Row reordering** is supported with Drag & Drop operations

Table column widths can be configured in percentage or pixels by admin. With percentage option, horizontal scrollbar will never be shown. If admin configures the table with "Cascade Delete" setting, all child relationships and related items are deleted when the context item is deleted. Admin can also configure specific column as "Freeze Column" to fix columns that are always visible and don’t move

along with horizontal scrollbar. Table can be configured to show either Scrolling or Pagination by admin.

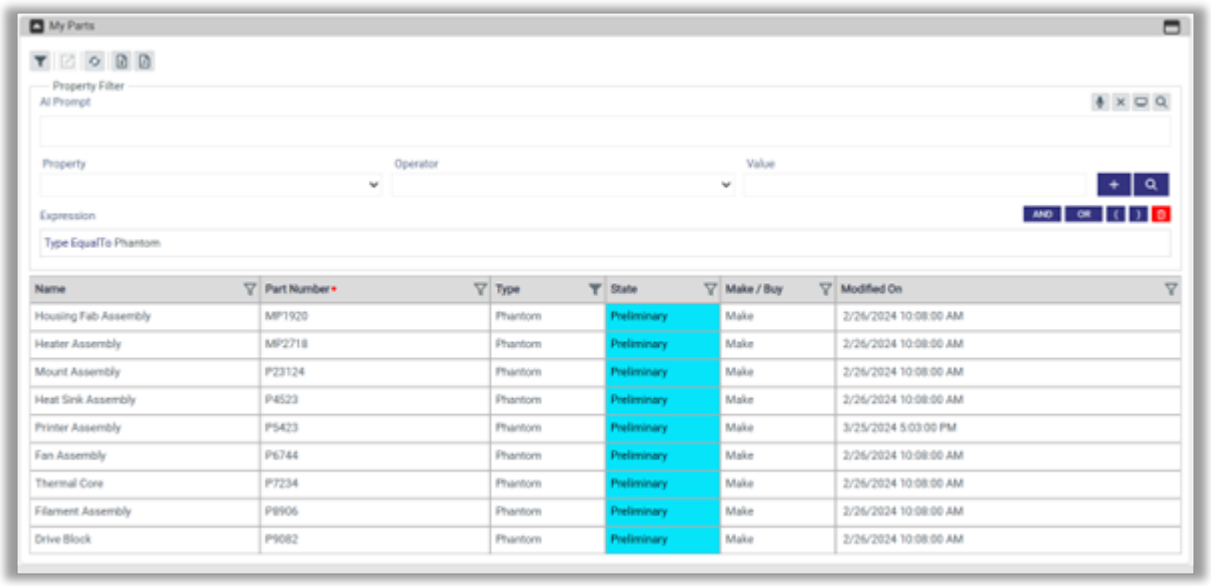


Worksheet Control

Worksheet control provides Excel experience for creating data sheets within Innovator based on ItemType data. Worksheet columns are defined based on ItemType properties and are constrained with property data types. It can show nested rows for a specific item with child relationships data. It has features like:

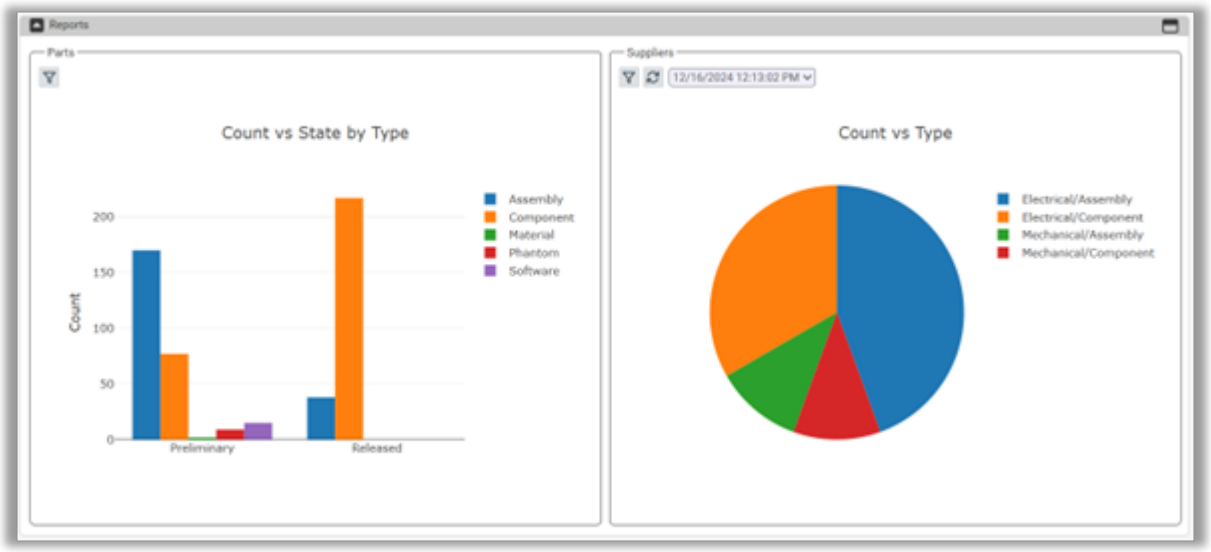
- **Computed Columns**,
- **Validation Rules**,
- **Conditional Formatting**,
- **Cell navigation using navigation keys** and all other shortcut keys are allowed for operations like Copy, Paste, Delete etc.
- **Excel type of sorting and filtering**. You can copy-paste between Excel and Worksheet control along with Drag & Drop operations for Row reordering,
- **Excel features** like Drag & Drop, Cell Copy by Drag and Undo & Redo etc.

Worksheet column widths can be configured in percentage or pixels by admin. With percentage option, horizontal scrollbar will never be shown. If admin configures the worksheet with "Cascade Delete" setting, all child relationships and related items are deleted when the context item is deleted. Admin can also configure specific column as "Freeze Column" to fix columns that are always visible and don’t move along with horizontal scrollbar.



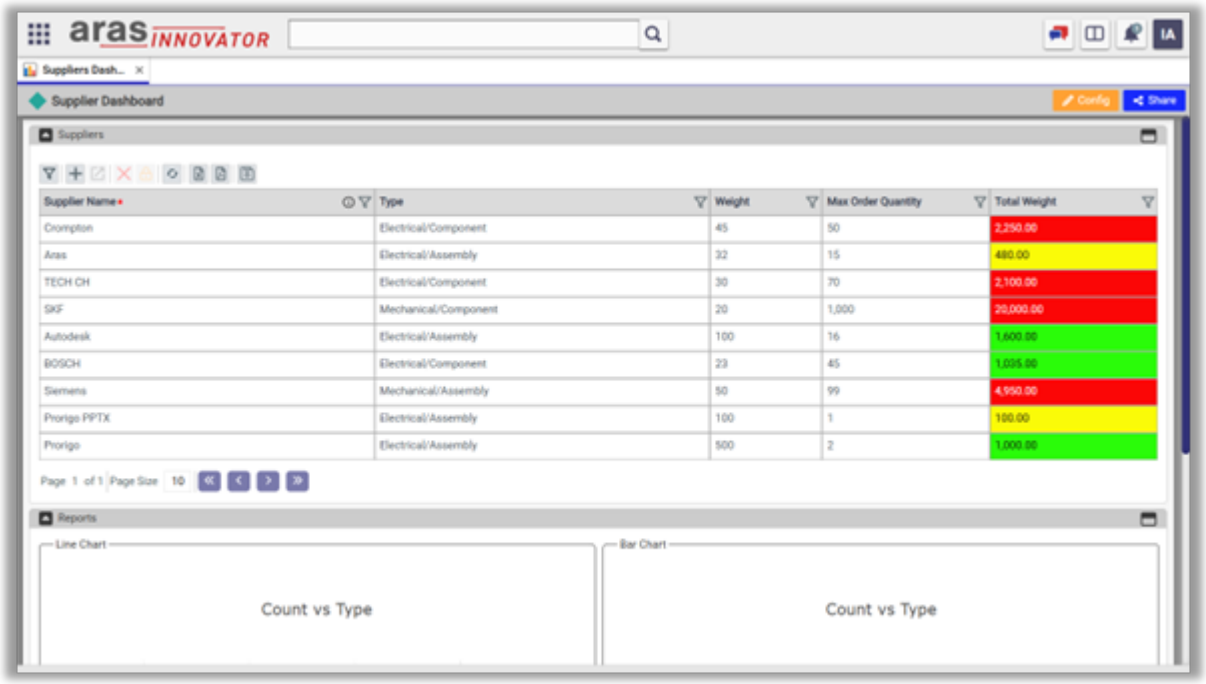
Report Control

Report control can be configured in Dashboard by the administrator using QueryDefinition. For Dashboards, report control should be configured with any QueryDefinition, which doesn't expect any item context. At runtime, report control executes QueryDefinition to get DataSource for the report. It can support any type of report like Pivot Table, Tree Table, Table Barchart, Heatmap, Row Heatmap, Column Heatmap, Line Chart, Bar Chart, Stacked Bar Chart, Area Chart, Stacked Area Chart, Pie Chart and PowerBI. All report types can show drilldown reports.



Type Dashboard

Type dashboard shows all the data related to an ItemType in the form of **table**, **worksheet** and **report controls**. You can have a table show list of items with pre-applied filter. You can also have reports generated based on that ItemType. Using Property filter, you can define complex filter criteria to fetch the items.



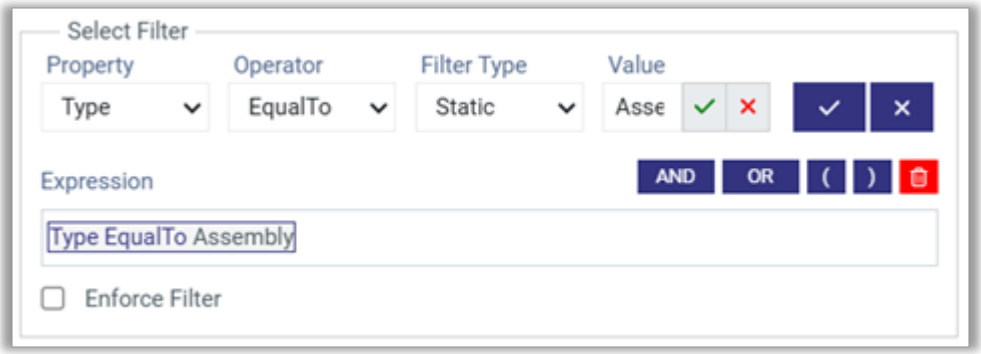
Property Filter

Property Filter is used in various controls of the Aras ProAppDesigner like Advanced Filter and Column Filter. Using Property Filter, you can create complex boolean expressions. Expressions can contain:

- Item properties,
- operators,
- values,
- grouping operator (‘( )’),
- logical operators (‘AND’, ‘OR’).

Various editors are used for showing values based on the selected property type:

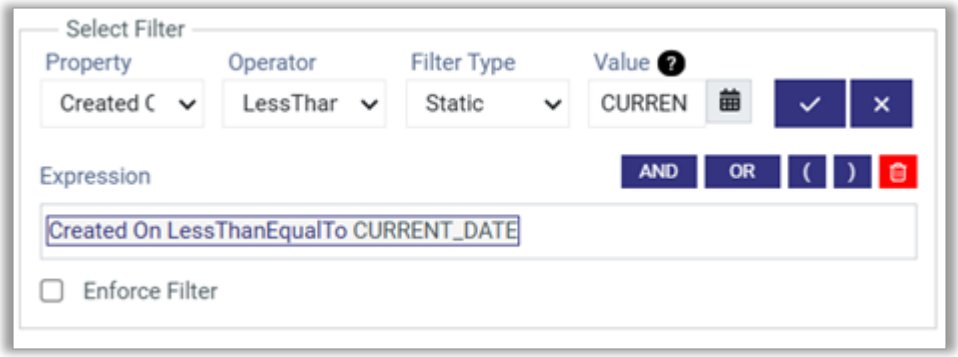
- Existing expression can be updated by selecting individual sub-expression or elements with mouse.
- You can also navigate expression elements using navigation keys. Selected keyword can be deleted by clicking on the delete icon.
- If no element is selected, by clicking on delete icon will remove the whole expression.
- If no element is selected in the expression, by inserting new property expression or other operators will insert at the beginning of the expression.
- If any element is selected in the expression, insert operation will insert the new element right after the selected element.



For the date property, Value field can take a specific date or a keyword. In order to build a boolean expression based on current date, value can contain keywords CURRENT_DATE and CURRENT_DATETIME. These keywords will be replaced at runtime before evaluating the expression with actual current date or datetime.

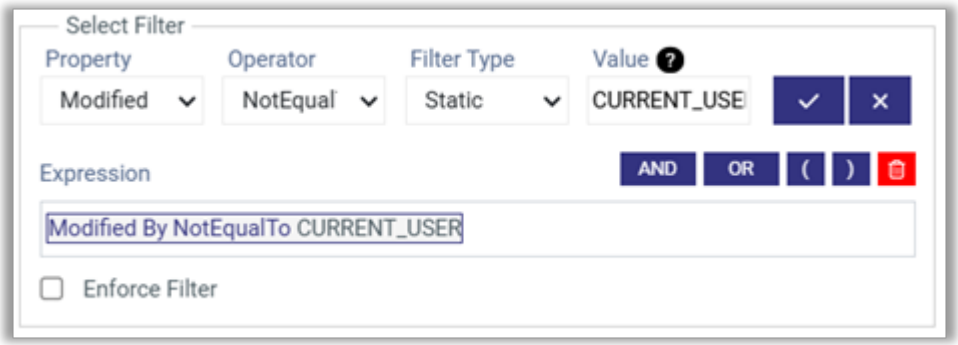
Ex1: **Created On** LessThanEqualTo CURRENT_DATE
Ex2: **Effective Date** LessThanEqualTo CURRENT_DATE + 2

Ex2: **Modified On** LessThanEqualTo CURRENT_DATE – 7



For the user property, Value field can take any text or keyword. In order to build a boolean expression based on current user, value can contain keyword CURRENT_USER. This keyword will be replaced at runtime before evaluating the expression with the logged-in user.

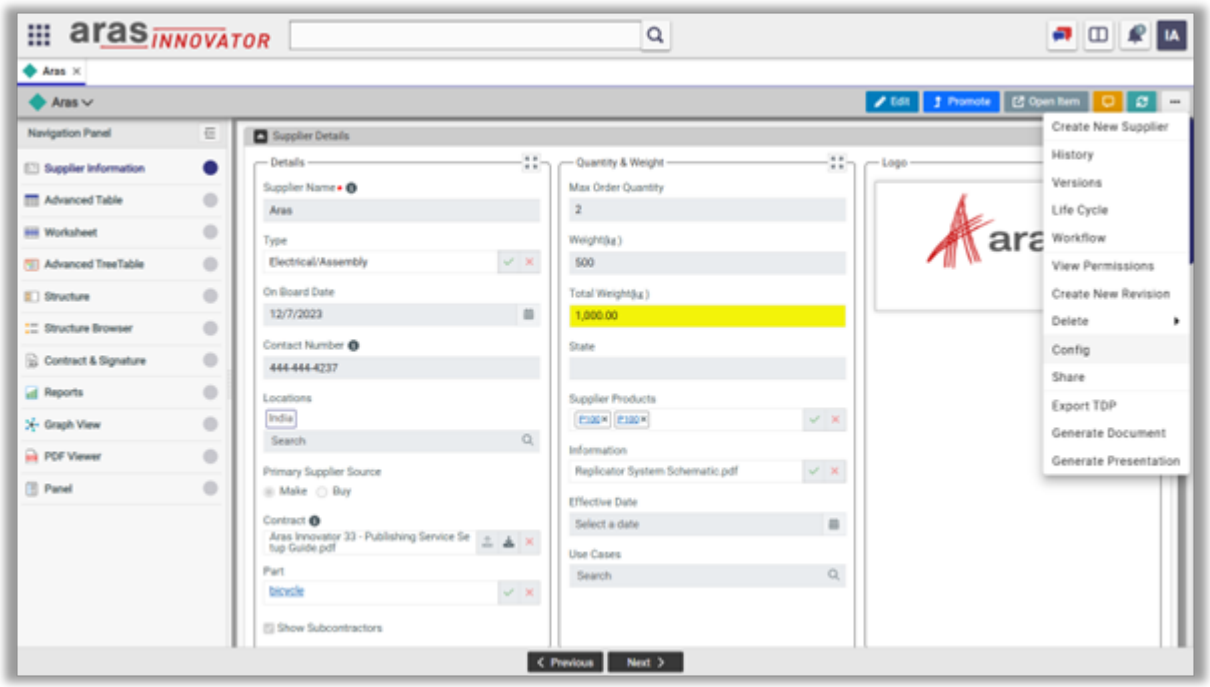
Ex1: **Created By** EqualTo CURRENT_USER
Ex2: **Owned By** NoteEqualTo CURRENT_USER



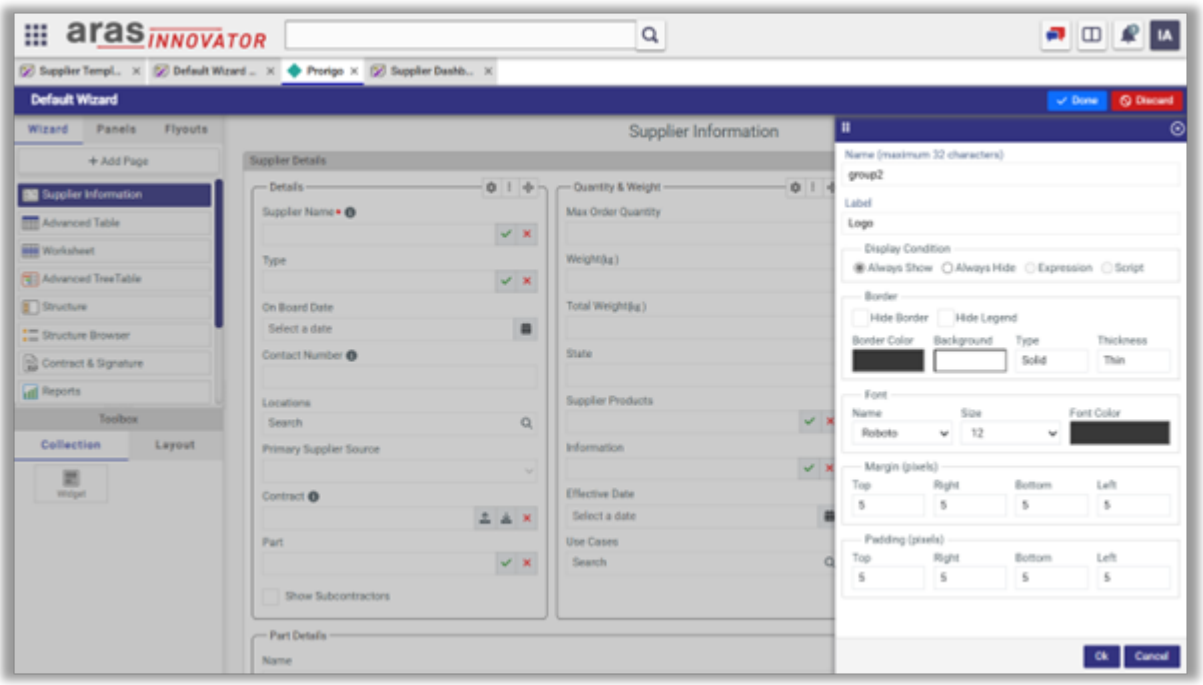
User Dashboard

When admin selects “Enable Config” option on template, end user can configure his view based on the **default view** shown to him at runtime. When admin selects “Enable Sharing” option, end user can **share his customized view** with other users in the in the organization.

End users can create their own views based on the admin defined wizards and dashboards. For each ItemType, end users can create any number of views for the wizard and the dashboard. These views are shown along with the default views defined by the admin. To create the view upon clicking Config action, end user will be shown with the designer using which he can configure or hide existing controls from the layout. He can also add new widgets from widget libraries to the layout.

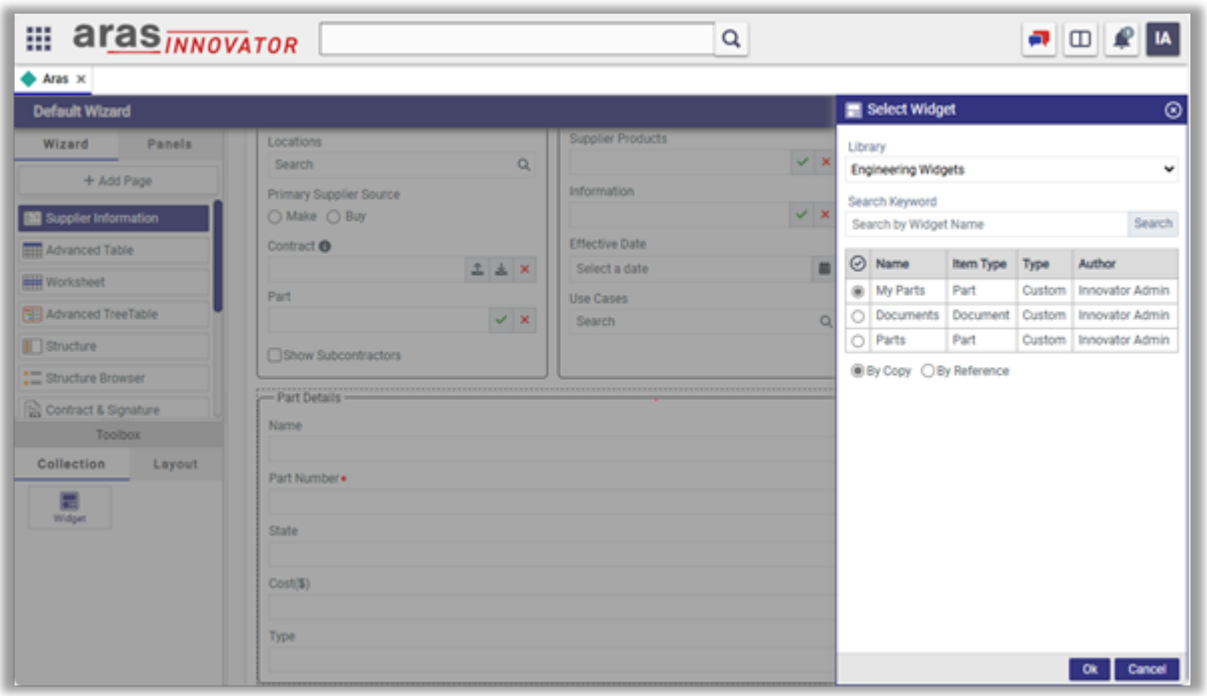


Upon selecting Config action from the dropdown, designer will be shown the current template, where end user can customize the view using designer capabilities. Using Display Condition option, end user can hide or show the control on the page. For collection controls like Table and Worksheet, by selection Settings icon, columns can be reordered or hidden from the view.

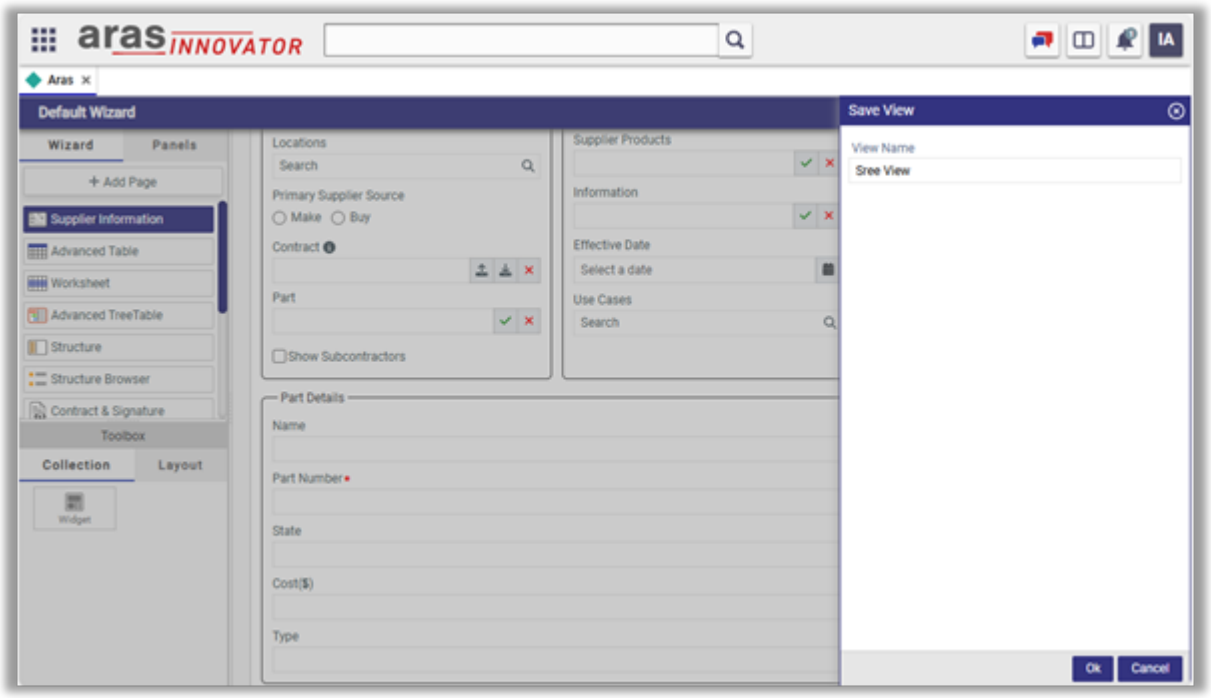


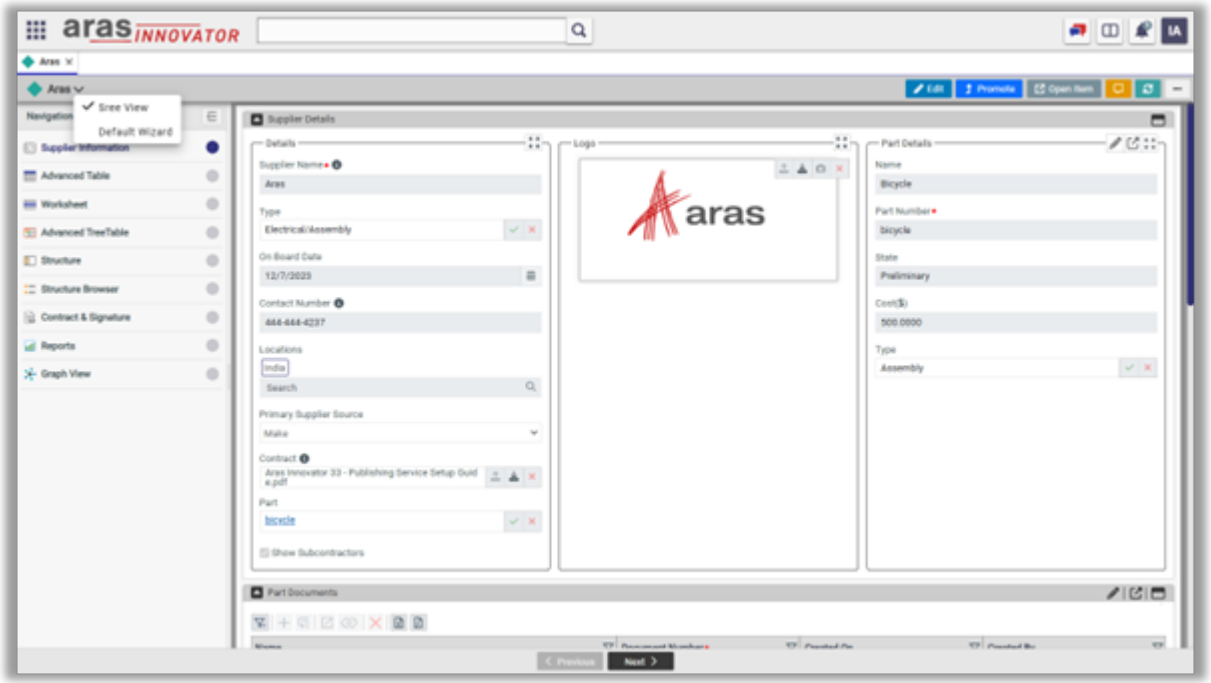
When admin creates any Widgets in the libraries, user can choose those widgets and place them on the page. Widgets are the predefined UI components, created from the existing collection controls of the template by the admin. Once defined can be used across many templates. Widgets are organized in libraries that are defined by admin. Admin can give access to the libraries based on user's roles. While creating wizard or dashboard template for a specific ItemType, admin can select a collection control and add it to the library.

Adding a widget from library to the template can be done using Widget control from the toolbox. Widget can be added to template by-copy or by-reference by selecting appropriate option from 'Widget Settings' flyout. Adding widget by-copy, will copy the definition of the Widget to the template. Any subsequent updates done on the widget in the library will not reflect in the template if the widget is added by copy. By adding widget by-reference, will always show the latest widget based on its updates in the library.

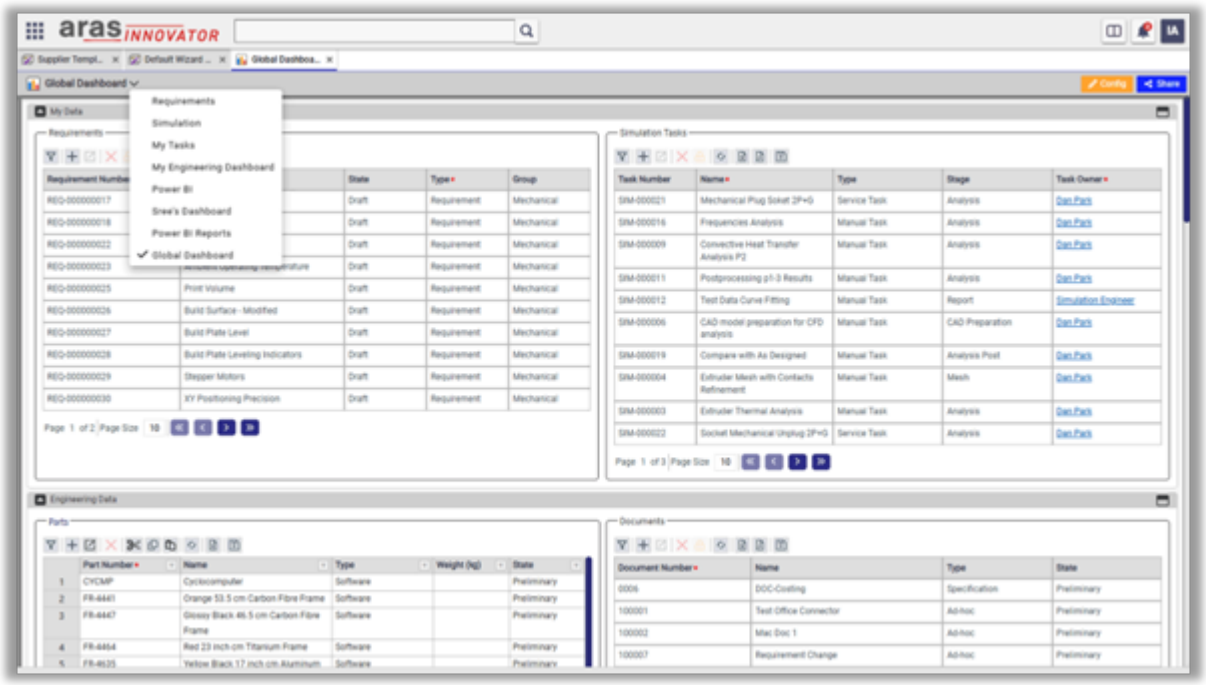


After making the changes, when user clicks on the Save action, flyout will be shown to enter name for the customized view. A new template view will be created with the given view name for the current user.





End users can create custom views for Global and Type dashboards similar to the Wizard.

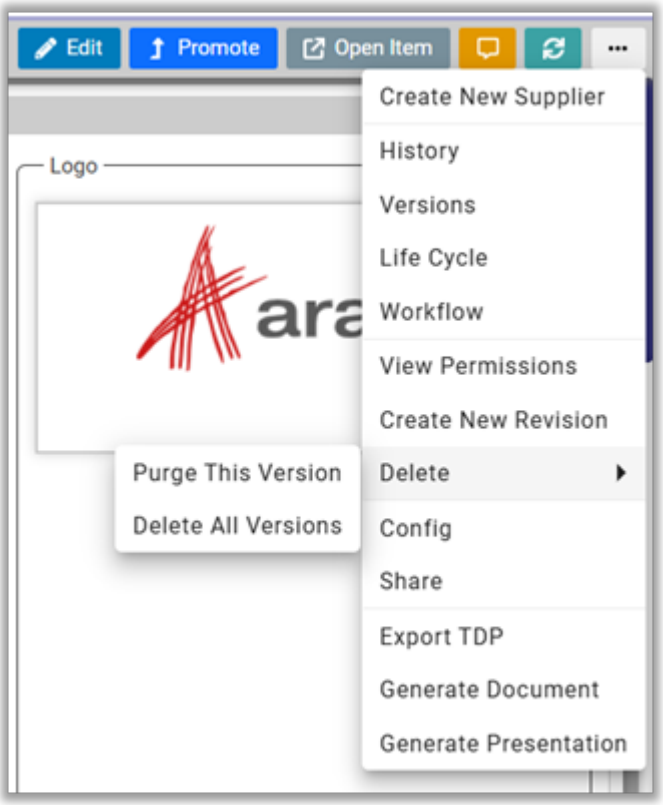


Wizard

Wizard is a separate single page application (SPA) which shows details of the item at runtime. When you select item from Innovator search grid, if the item has published Aras ProAppDesigner template, item details with be shown using the template instead of using default Innovator form. When you click on Create action from search grid, same template will be used to show the page with empty controls.

Wizard Actions

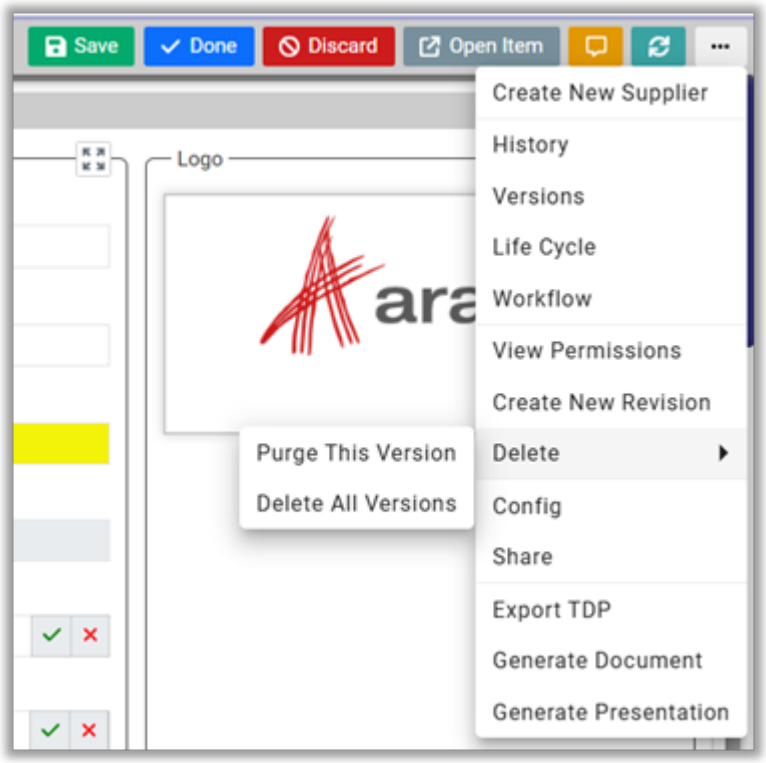
Wizard actions are shown in the wizard in the top-right corner when the item is opened. When you open item in read-only mode, you will see actions like Edit, Promote and Open Item.



- **Edit**: Applies lock on the item with exclusive access to make changes on the item.
- **Promote**: Promotes the item to the next state of the lifecycle associated with the item.
- **Open Item**: Shows new Innovator tab with default form that shows item details in classic view.
- **Discussion Panel**: Shows discussion panel as a flyout with all messages and their replies exchanged in the current item context.
- **Refresh**: Reloads the current item by making server request.
- **Create New Item**: Shows new item page in a different tab.
- **History**: Shows history of all the changes made to current item.
- **Versions**: Shows list of all previous generations of the current item.
- **Life Cycle**: Shows associated life cycle of the current item.
- **Workflow:** Show associated workflow of the current item.
- **View Permissions**: Shows permissions associated with the current item.
- **Create New Revision**: Creates a new revision for the current template item.
- **Delete**: Allows to delete current generation or all generations of the item.

- **Config**: Config action is shown only when admin has configured the template view in order to allow end user to customize the layout of the wizard of a given ItemType.
- **Share**: Share action is shown only when admin has configured the template view in order to allow end user to share the layout of the wizard of a give ItemType with other users.

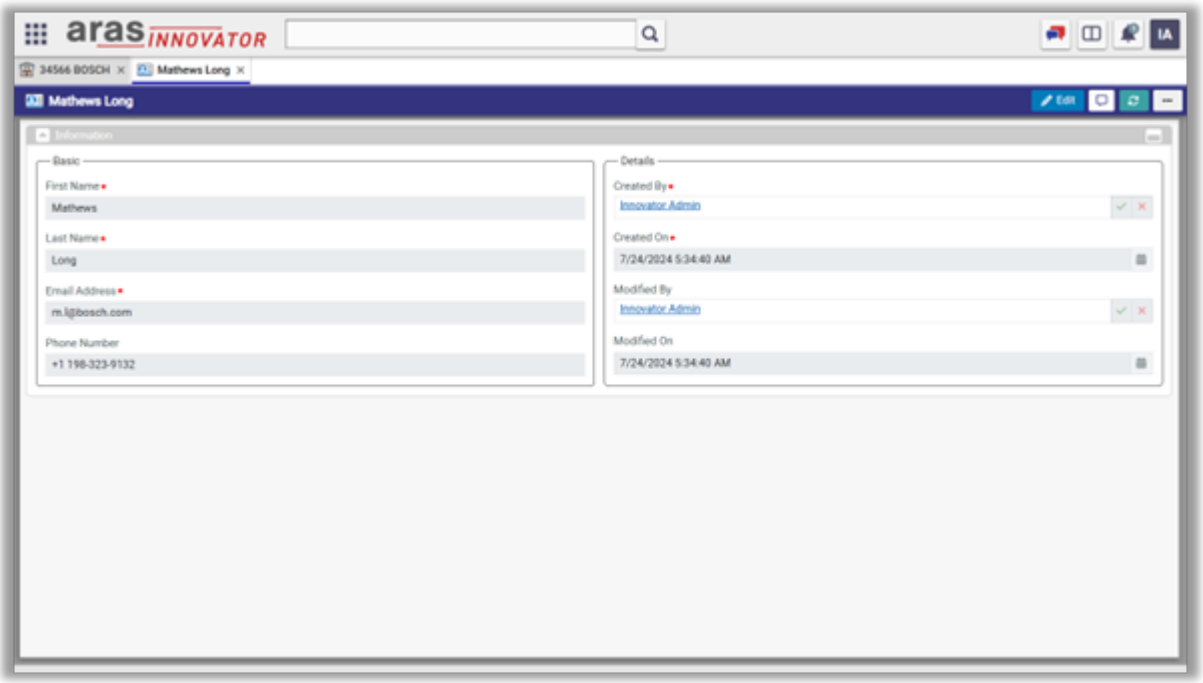
When you open the item in edit mode, following actions will be shown.



- **Save**: Saves the changes done so far on the wizard. Once changes are saved, you can't discard them using Discard action.
- **Done**: Saves the changes done so far on the item and unlocks the item so that other people can make changes.
- **Discard**: Discards all the changes done so far on the item after the last save and unlocks the item so that other people can make changes.

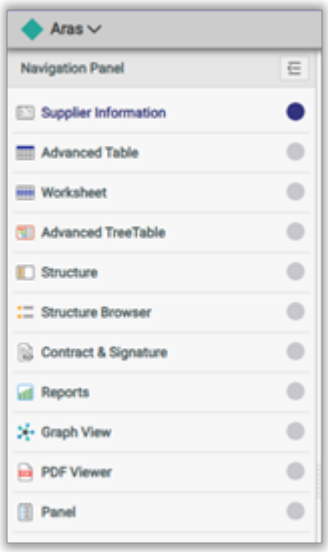
Single Page

The template created in Aras ProAppDesigner can be single page or multi page. If the template has only one page, then it is called single page application. In case of single page wizard, item keyed-name will be used to show the page title.



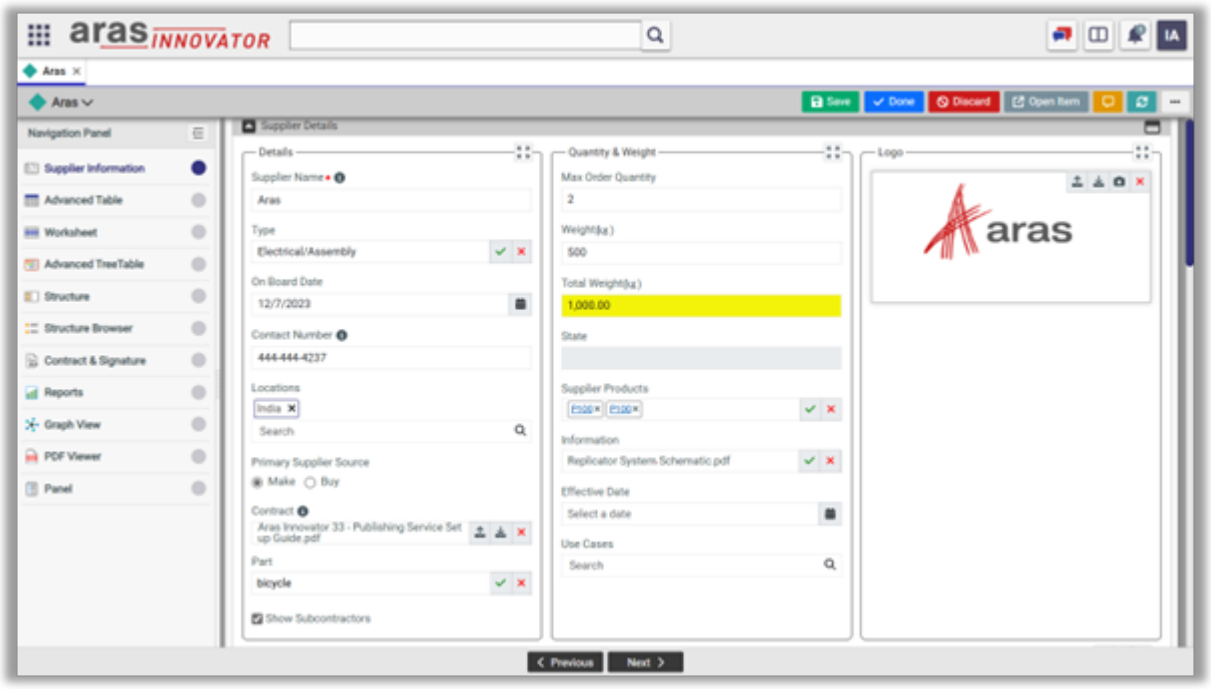
Multi Page

If the template created in Aras ProAppDesigner has more than one page, then it is called multi page application. Multi page application is shown with Navigation panel on the left which displays all the pages of the application. Active page is shown with blue-circle icon, already visited or filled pages are shown with green-circle with tick icon. If the page is not visited, then it will be shown with gray-circle with tick icon. If the page is visited but there are validation errors, then it will be shown with red-circle with tick icon. Along with the Navigation panel, it shows the Navigation Bar at the bottom with buttons Previous and Next. Previous button is disabled when you are on the first page and Next button is disabled when you are on the last page. Sometimes, all the pages may not be shown on the Navigation panel if those pages are defined with Display Condition. Display Condition expression is evaluated every time there is a change in contained property values.



Page Navigation

- You can navigate through the pages by clicking Next and Previous buttons in the bottom Navigation Bar. You can navigate to next page by clicking Next button, as part of moving away from the page, all validation rules on the page are evaluated. If any of the rules are not satisfied, it shows popup dialog with Validations Failed message and icons against each control.
- By hovering the mouse on the icon, specific errors or warning messages are shown based on the defined validation rules on the control. You can move away from the current page by clicking any other page from the Navigation Panel. If the current page has any validation errors, a popup dialog will be shown with options to ignore and move forward or fix the issues before moving forward.
- Every time when you save the item if you have “as_is_validated” property on the ItemType, it will save the status of validations against that property. That property value will be set to true if all pages in the wizard have no validations errors. Even if a single page has validation errors, property value will be set as false.



Page Controls

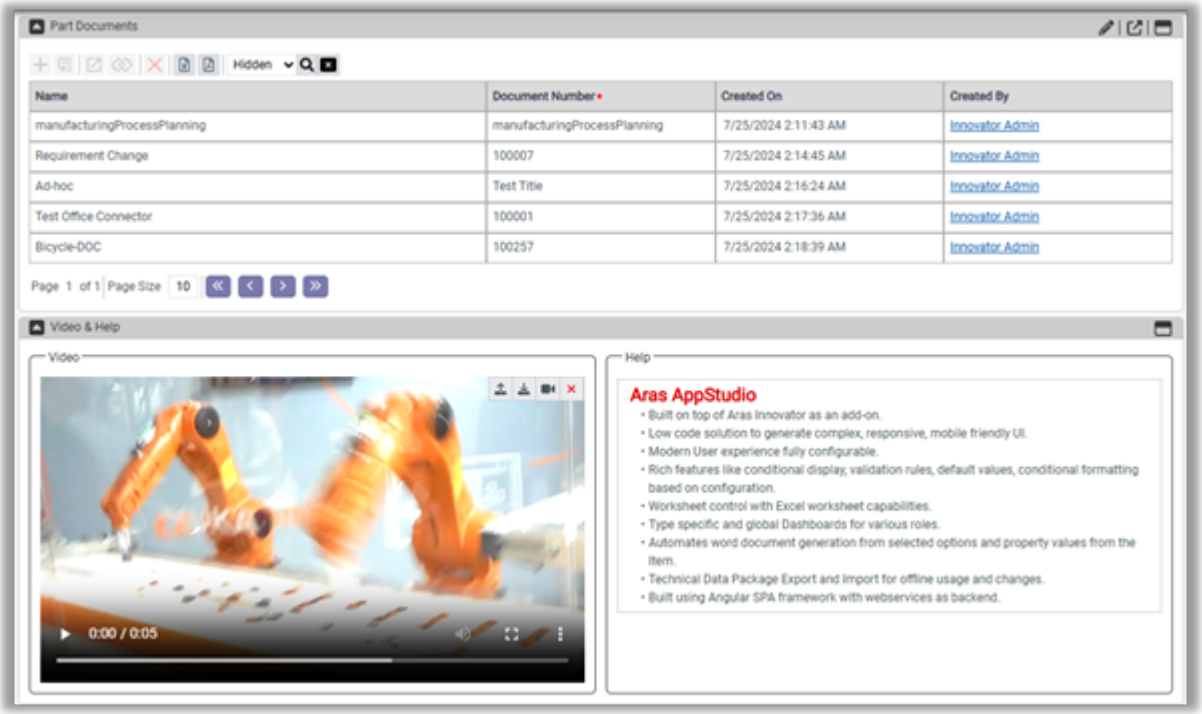
Every page is composed of layout controls, simple controls, collection controls. You can also define interdependencies between controls using features like **Display Conditions**, **Validation Rules**, **Conditional Formatting** and **Reset Dependencies**; these topics are covered later in the document.

Layout Controls

Layout controls can organize simple and collection controls with a proper layout in the page.

Section

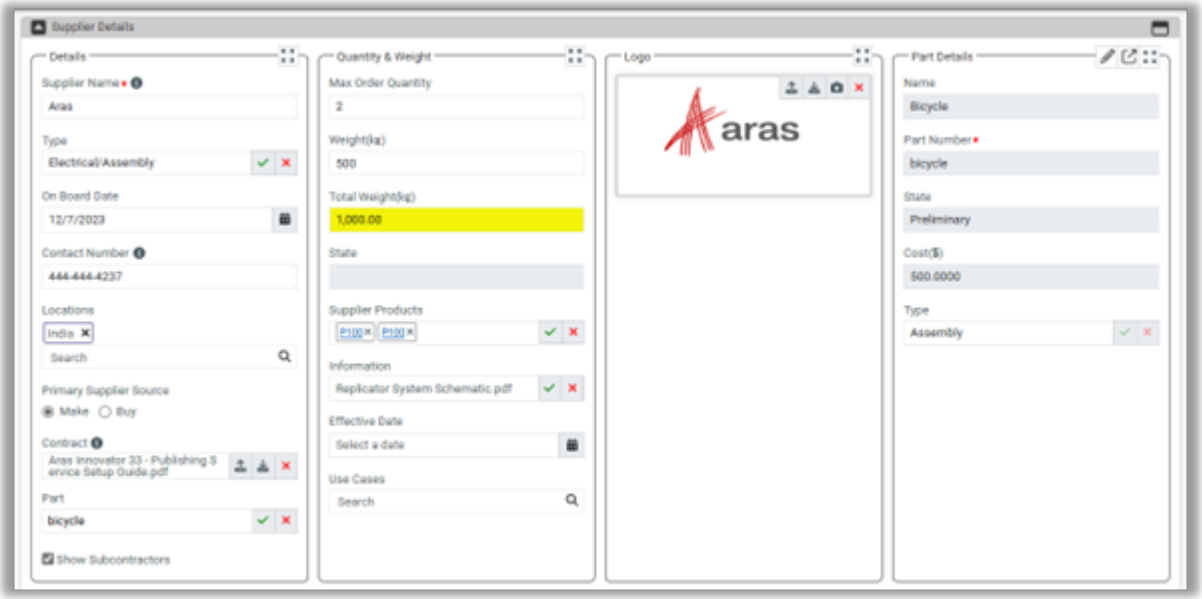
Section control is a layout control to organize group controls placed inside it. Section controls are vertically stacked one over the other. Section controls are the top-level container controls exists in pages and flyouts. By defining display condition or access permissions, section control can be shown or hidden based on the evaluation criteria. In the picture below, Part Documents and Video & Help represents sections in the page.



Group

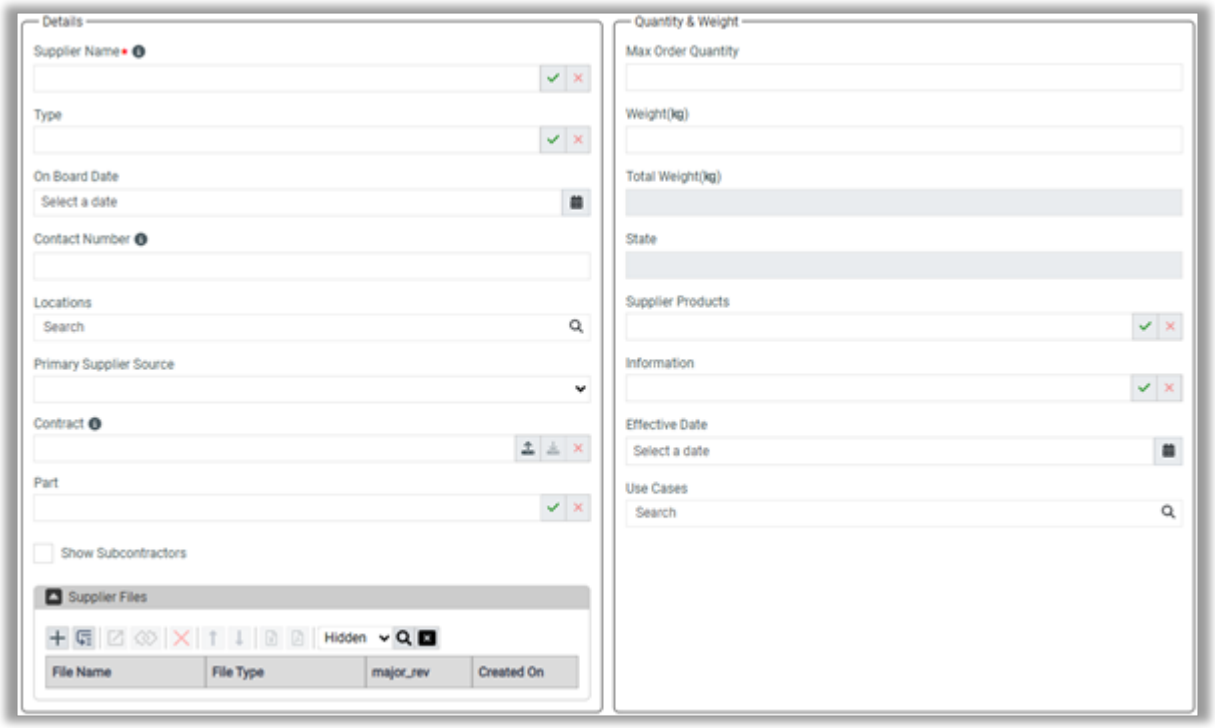
Group control is a layout control to organize child controls placed inside it. Group controls are horizontally stacked beside each other inside a section control. When multiple group controls are placed inside section control, they equally take the available width. When you resize the page on desktop, based on the available width they can be shown in single row or multiple rows inside a section control. In mobile view, they will be vertically stacked inside a section control. By defining

display condition or access permissions, group control can be shown or hidden based on the evaluation criteria. In the picture below, Details, Quantity & Weight, Logo, Part Details represents groups within ‘Supplier Details’ section.



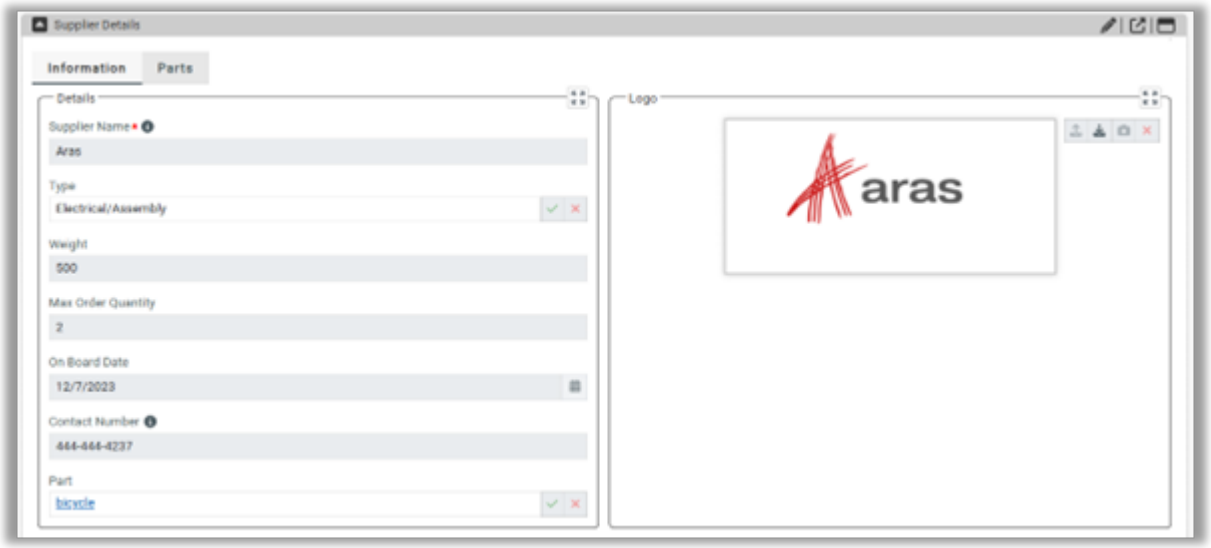
Panel

Panel control is the layout control that can help you to create a complex layout on the page. Panel control can be created just like a page with section controls within it. Panel control can be placed within group control just like any other child control. It can also be used as tabs within Tab control. In picture below, ‘Supplier Files’ represents panel control, that is placed inside the group.



Tab

Tab control can be used to show associated panels as tabs. In the picture below, Information and Parts together represents Tab control built based on the panels.

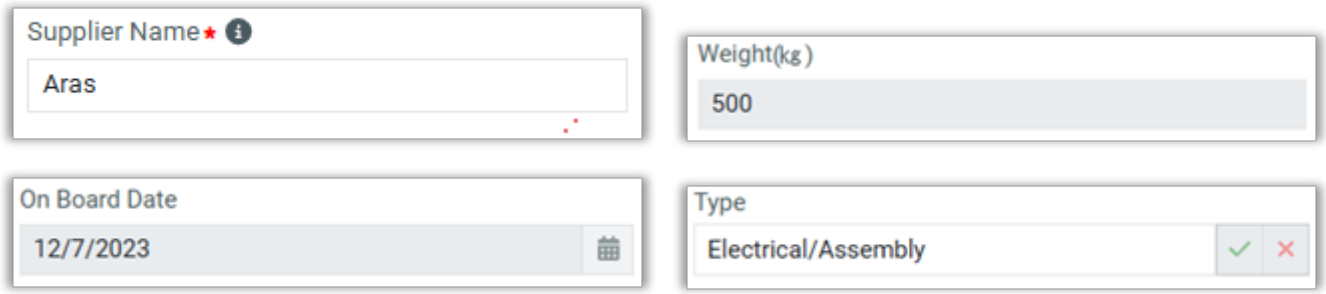


Simple Controls

Simple controls can show data related to an item or its properties.

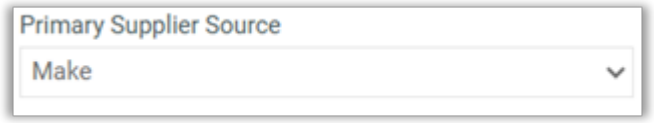
Text

Text control is used to display properties of datatypes String, Text, Integer, Float, Decimal, Date, Sequence and Classification. Text control honors Format Specifier set on the ItemType definition. Text control honors the default value set through ItemType definition. You can also set default value for Text control by defining text expression with properties from other controls from the current page or previous pages. Text control has additional settings like Display Condition, Validation Rules, Conditional Formatting etc. that are covered in the subsequent sections.



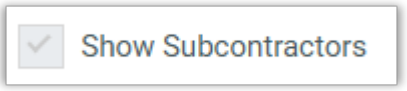
List

List control is used to display properties of datatypes List, Filter List and Multi Value List. While displaying list, Aras ProAppDesigner uses same underlining list with options defined in the Innovator. List control comes with Control Style setting, with this, options can be shown as RadioButtonGroup in case of single selection list or CheckboxGroup in case of multi selection list. If the Control Style is not set, default Dropdown will be used to show the options. List control honors the default value set through ItemType definition.



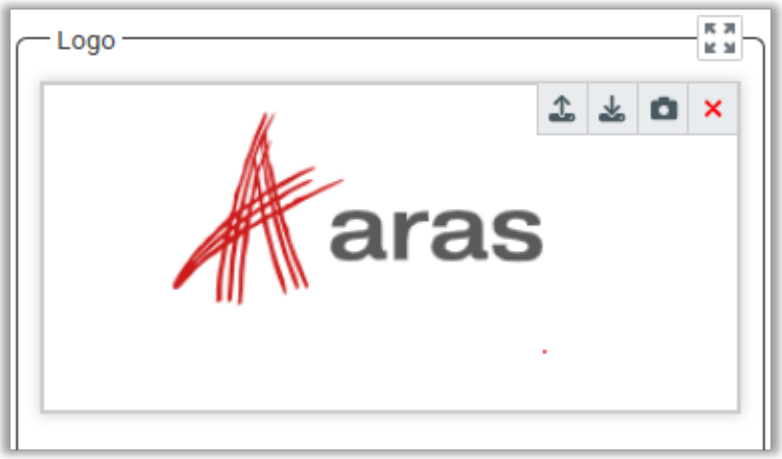
Checkbox

Checkbox control is used to display properties of datatype Boolean from the associated ItemType.



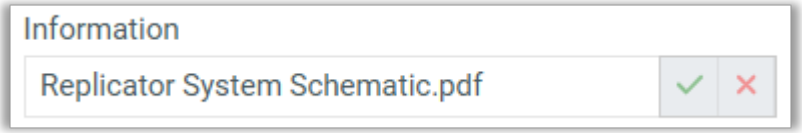
Image

Use Image control to display properties of datatype Image from the associated ItemType. Image control will show actions to upload picture, download picture, capture picture, and remove picture. Capture picture can be used to capture picture directly from the camera while using the application in the mobile device.

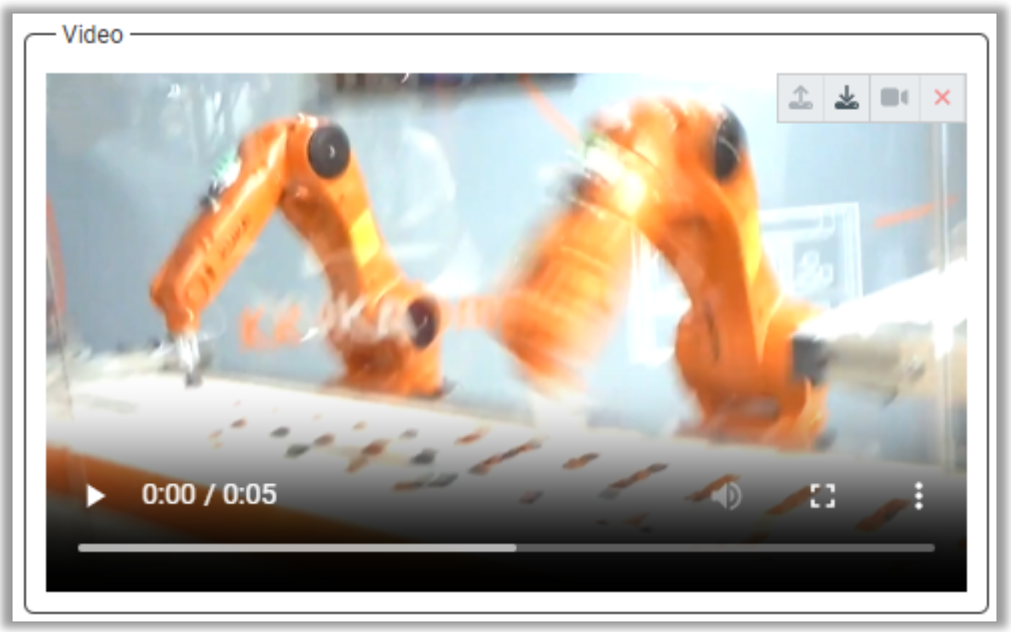


File

Use File control to display properties of datatype File from the associated ItemType. Admin can setup the file control in File or Video mode inside template definition.

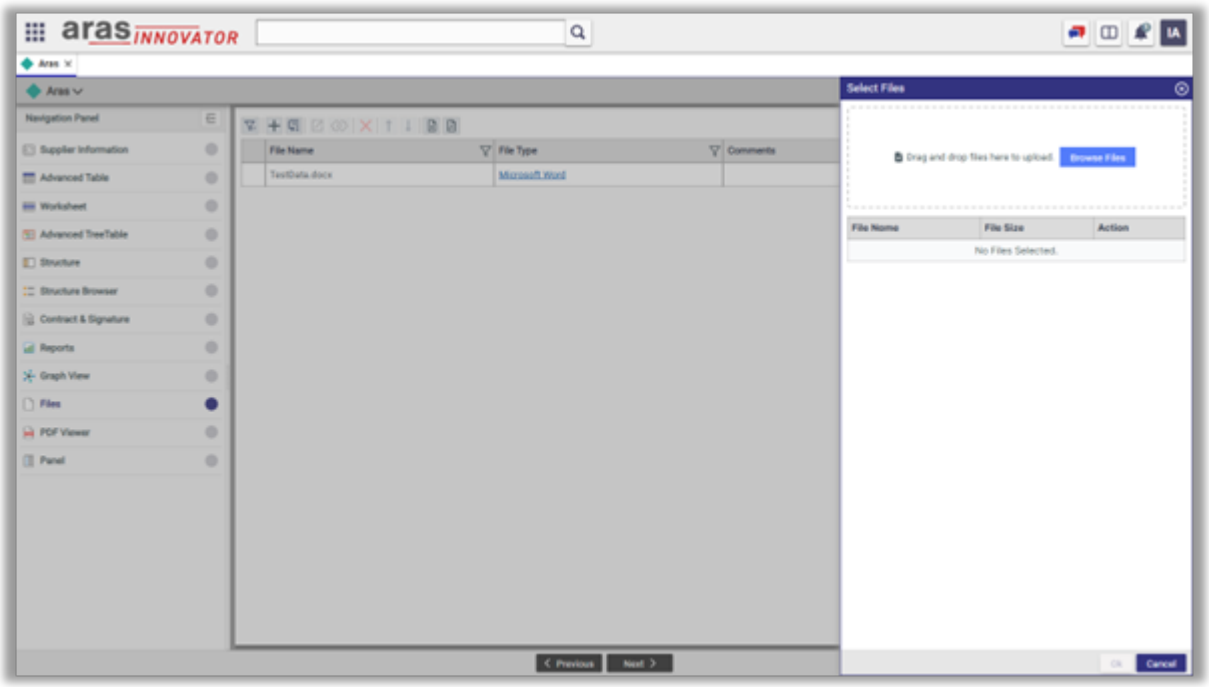


When it is set in video mode, it will show actions to upload video, download video, capture video, and remove video. Capture video can be used to capture video directly from the camera while using the application in the mobile device. It also has inbuilt support to play the video.



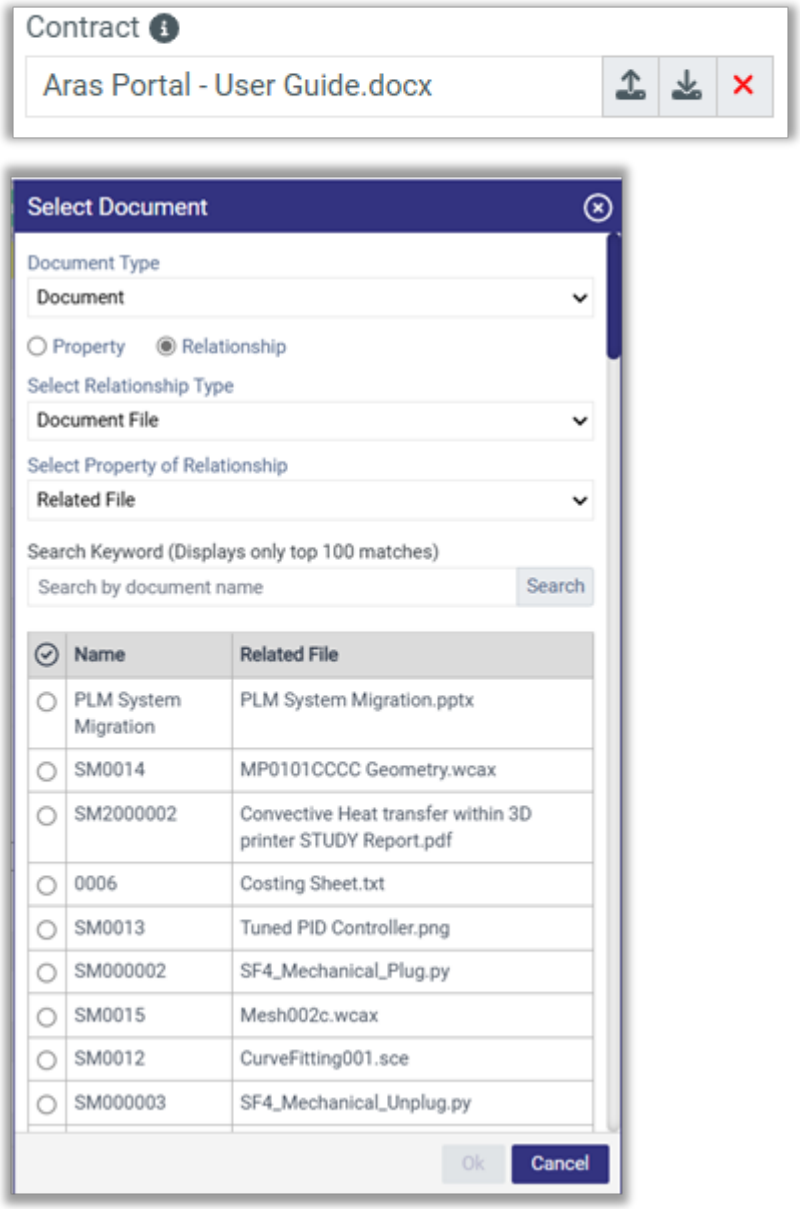
File control on the form support drag & drop of files from the local file system. When you drag & drop a file from the windows explorer to file control on the form, it will show dropped file path in the control.

File property used in collection controls columns shows a file selection icon in the column header to support bulk file upload in the collection control. Upon selecting the icon shows ‘Select Files’ flyout with drop area. It also shows ‘Select Files’ flyout when you select Add action from Files table.



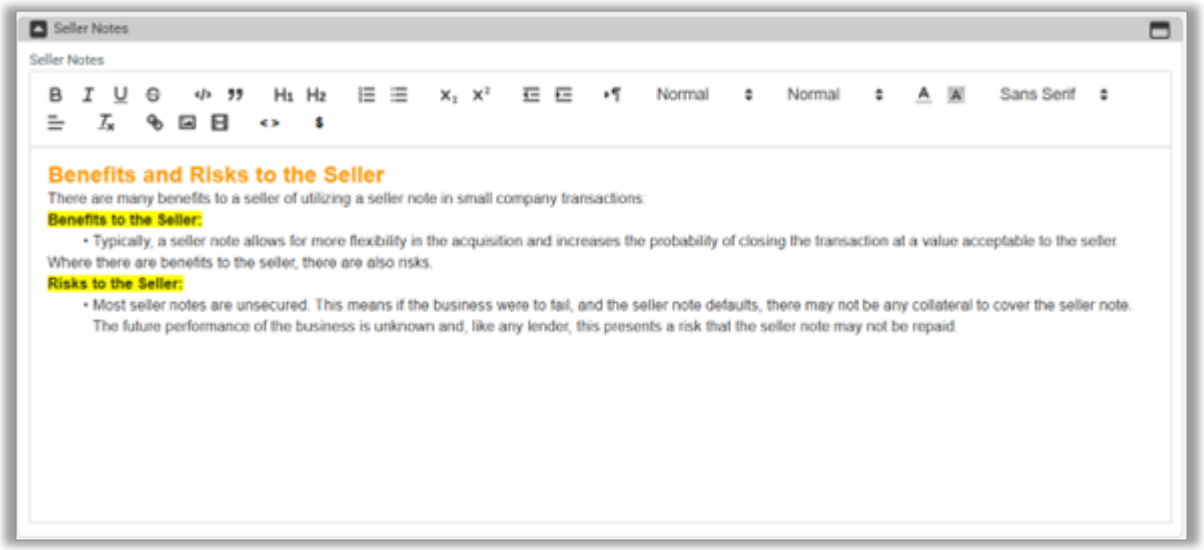
Document

Document control allows you to select the document which is stored in Innovator. Upon clicking on tick icon of the control, Select Document dialog will be shown. After entering search criteria, Select Document dialog shows all the files of type Word, PDF and Image. Selected document is stored on the context item as file property. Document control is useful in case you want to generate Word/PDF document from the context item.



RichText

RichText control displays properties of datatype FormattedText from the associated ItemType. In edit mode, RichText control shows toolbar, using it you can format the text with bold, italic, font, color etc. It also allows you to insert items like images, hyperlinks and videos, upon saving all these items are embedded inside FormattedText on the property.



xClassification

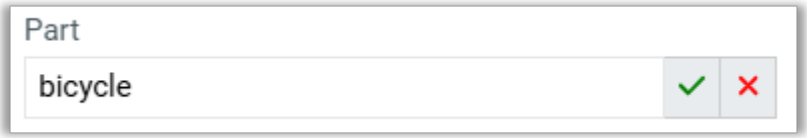
xClassification control associates xClassification tree nodes and xProperties with the item. It works exactly same way as you work with regular Innovator forms.

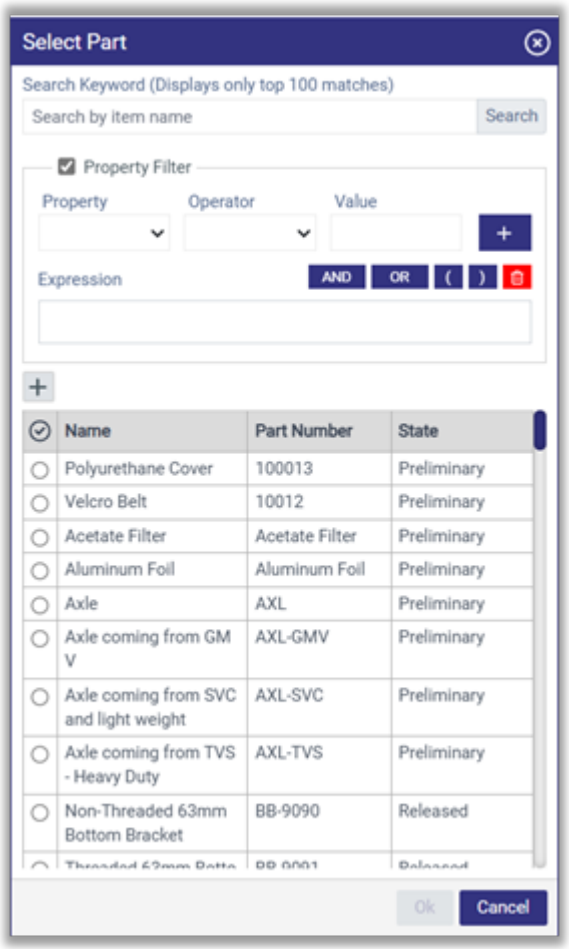
Signature

Signature control captures your digital signature as image and can be saved to image property of the context item. Signature control shows a signature pad, you can use mouse to draw your signature.

Item

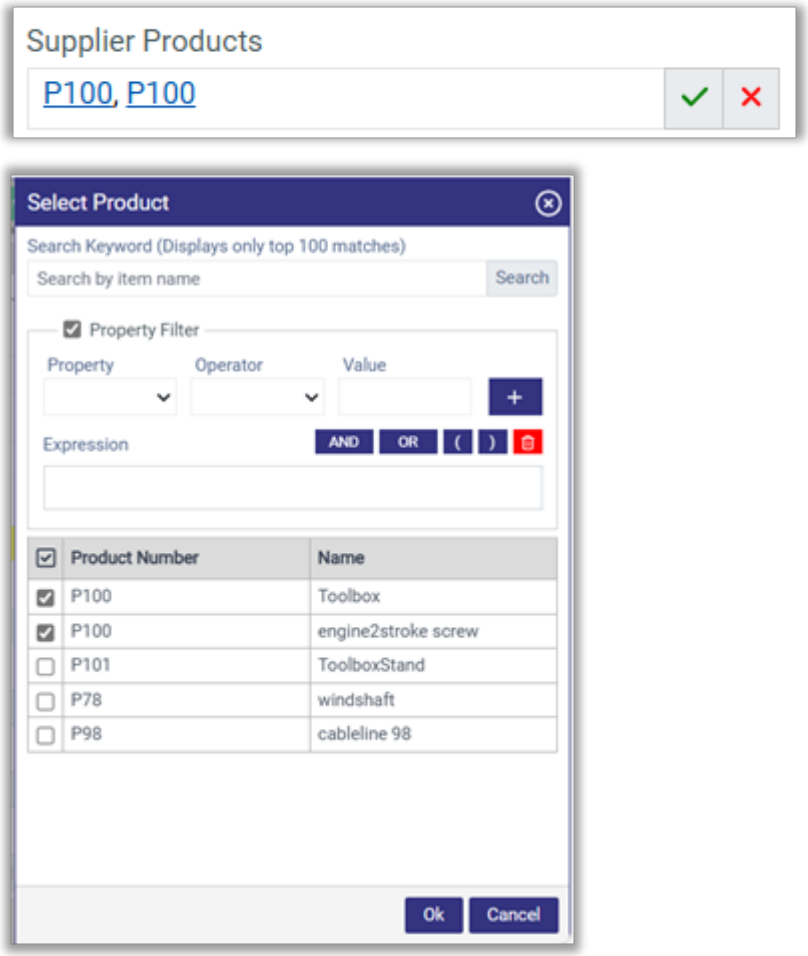
Item control displays properties of datatype Item from the associated ItemType. Using this control, you can select existing item of the property's Data Source ItemType through Select Dialog.





Items

Use Items control to select and display list of items with their keyed-names from the associated ItemType. Using this control, you can select existing items of the property's Data Source ItemType through Select Dialog. Selected items will be saved as relationships upon saving the context item.

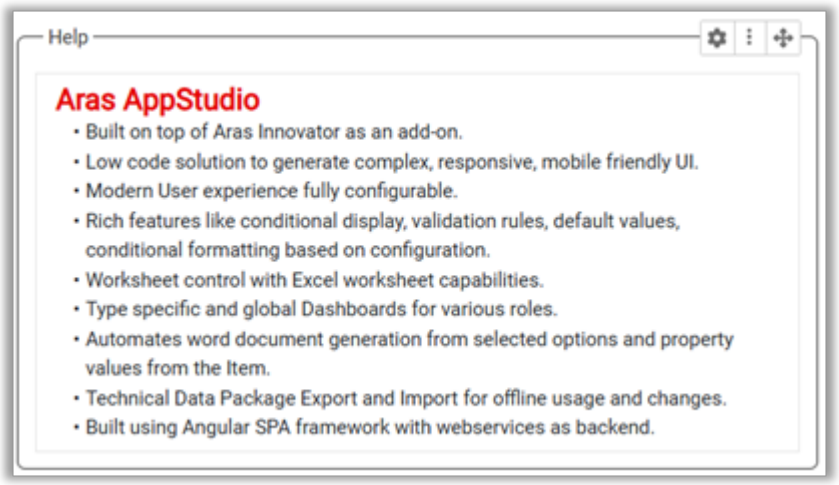


ButtonGroup

ButtonGroup control displays a group of buttons or a single button. Button control can be associated with a custom action defined by admin. Each button can be configured to show a dialog or a shortcut to the existing Innovator page. Button can be also be configured to call a custom script.

Banner

Banner control shows static content like instructions or help on the page. Banner control with static content is added through rich-text editor inside the template. At runtime, it shows rich-text to the user.



Frame

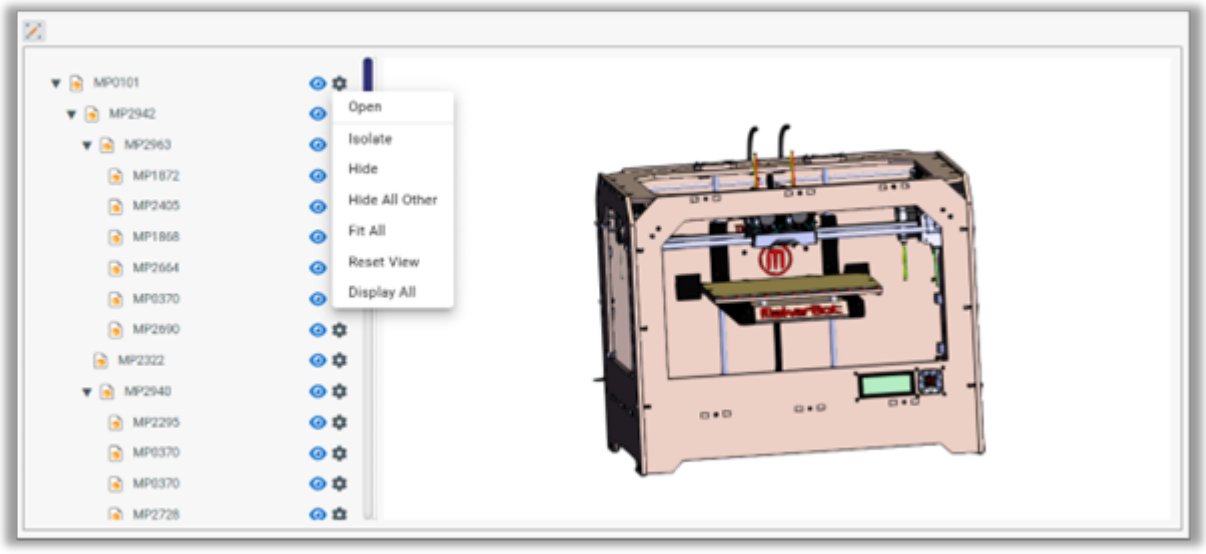
Frame control displays content from the external page. It can be used inside the page to show a report generated from the external BI tool.

HTML

HTML control displays custom UI in the pages and dialogs. HTML content added to this control will be rendered in the same DOM as of the Aras ProAppDesigner Wizard.

CAD Viewer

3D Viewer displays 3D Models developed on software related to CAD. It provides the external users with a detailed view of the item. 3D Models are shown on the right panel and a Tree View of all the individual items is shown on the left. 3D Models can be moved in any direction and can be zoomed in or out, by using the basic mouse and touch (mobile) actions.



Toolbar Options:

In the Tree View each tree item has an option to show or hide it, along with other options stated below:

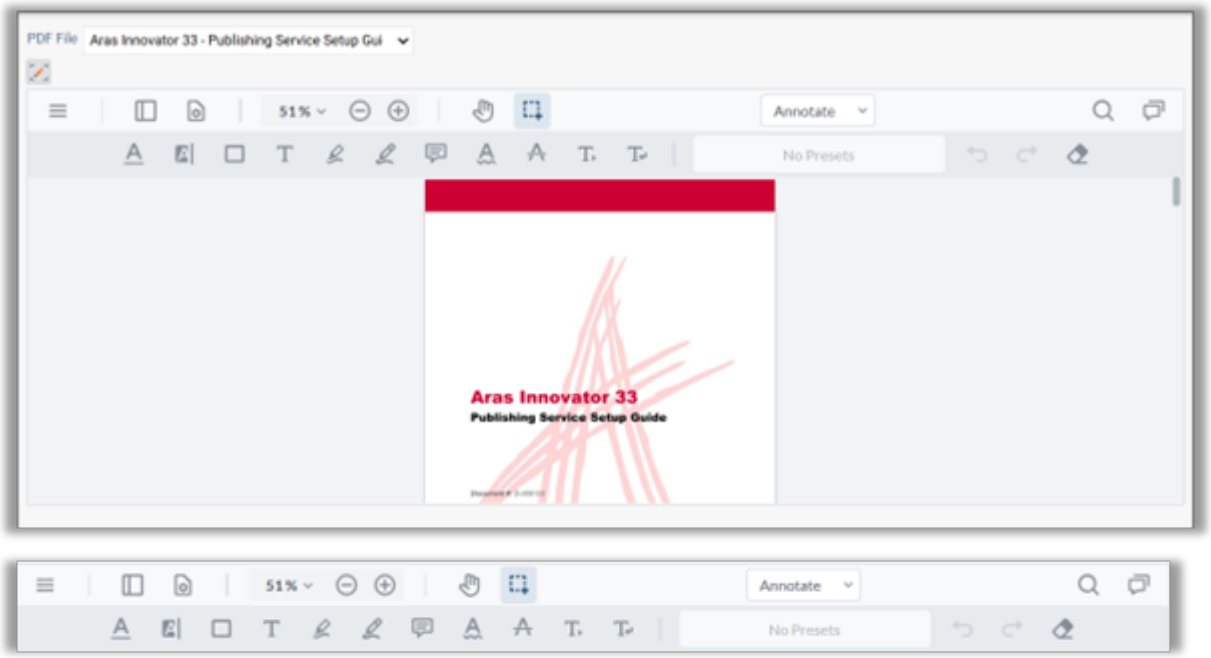
- **Open**: Opens a new tab for the individual item, to access all its information.
- **Isolate**: Focuses and shows the current item, by decreasing the opacity of all other items.
- **Show / Hide**: Shows or hides the current item.
- **Hide All Other**: Hides all items excluding the current item.
- **Fit All**: Resizes the 3D model based on the size of the screen.
- **Reset View**: Reloads the 3D model.
- **Display All**: Shows all items .

Markup:

Click on the Markup button to take a snapshot of the 3D model view and open it in the Image Viewer. You can perform all the basic image editing actions and download the image from the Image Viewer. You can return to the 3D Viewer by clicking the top left button of the Image Viewer.

PDF Viewer

PDF Viewer allows to view 3D PDF and PDF documents associated to that item. PDF Viewer consists of a Toolbar with Panel, View Controls, Zoom, Pan and Select options on the left. While Search and Menu options are on the right



Toolbar Options:

All Toolbar options are briefly explained below.

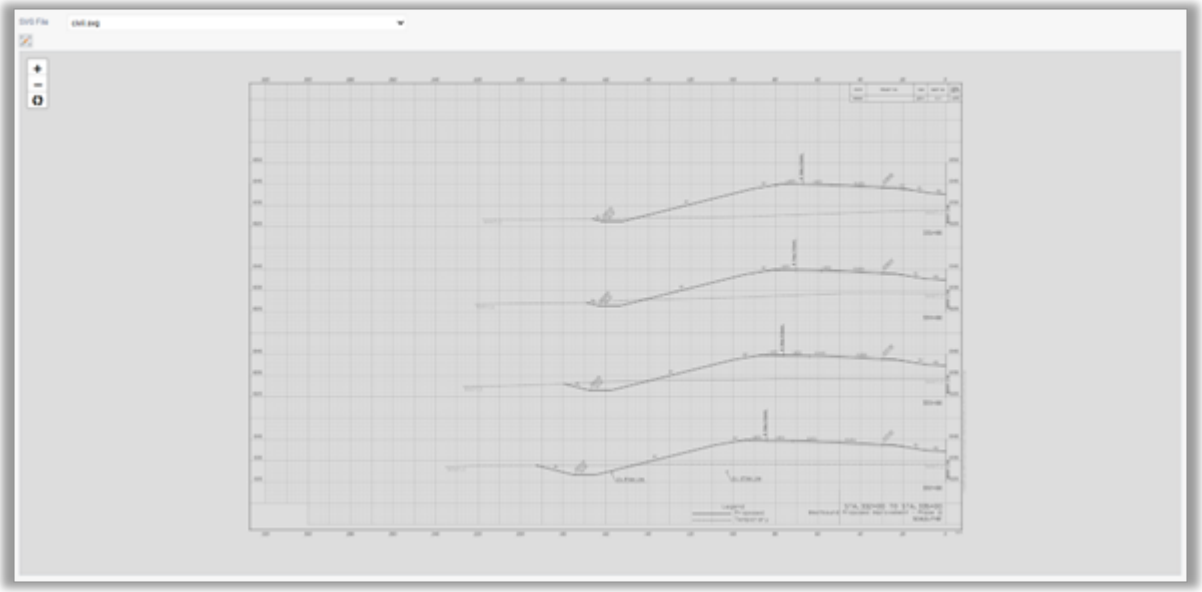
- **Panel**: Opens left panel showing preview of all the pages in the PDF, can be used to move across various pages.
- **View Controls**: Provides options to change the Transition, Orientation & Layout of the page view.
- **Zoom**: Provides options to Zoom in, Zoom out, Fit to width, Fit to page, Marquee Zoom and other fixed Zoom options.
- **Pan**: Moves the models or pages.
- **Select**: Selects text or models in the page.
- **Search**: Opens a right panel where user can perform search operations on the PDF document.
- **Menu**: A dropdown shows up containing the options below:
  - **Full Screen**: View the PDF in full screen mode.
  - **Download**: Download the PDF document.
  - **Print**: Take a printout of the PDF document.
  - **Dark / Light Mode**: Change the theme of the viewer.
  - **Languages**: Change the language of all the options available in the viewer.

Markup:

Click on the Markup button to take a snapshot of the page view and open it in the Image Viewer. You can perform all the basic image editing actions and download the image from the Image Viewer. You can return to the PDF Viewer by clicking the top left button of the Image Viewer.

Image Viewer

Use ImageViewer control to render SVG and image files associated with a Document item. Image viewer show list of all files in the dropdown associated with document item. It also shows controls to zoom in, zoom out, and refresh.



Markup:

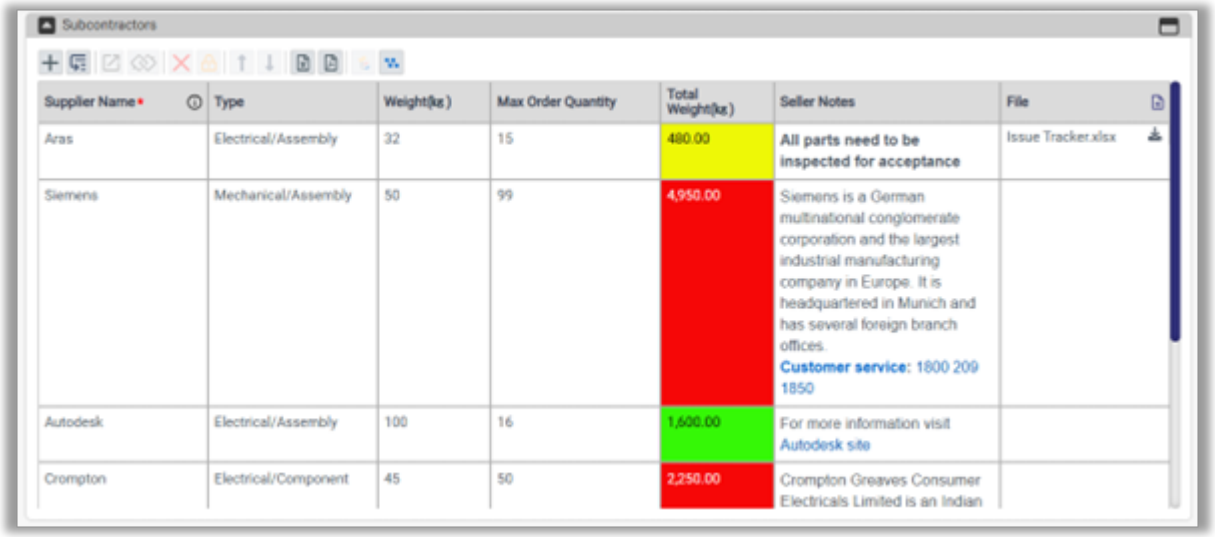
Click on the Markup button in the toolbar to take a snapshot of the image and open it in the editor. You can perform all the basic image editing actions and download the image from editor. You can return to the Image Viewer by clicking View button from the toolbar.

Collection Controls

Collection controls can show data related to collection of items and their relationships.

Table

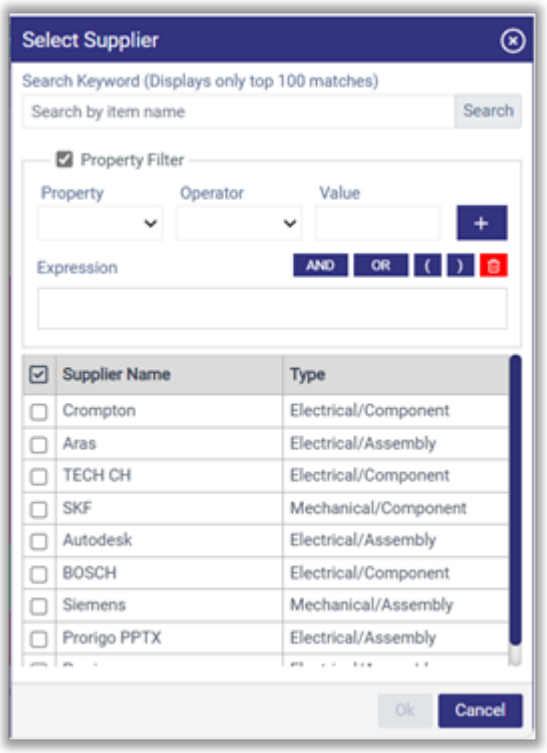
Table control shows relationship or reference data of the context item. At the time of adding table control to the form, properties of RelationshipType or ReferenceType are mapped to table columns. Table control has multiple settings that control its appearance and behavior at runtime.



Above the table control, on the left end you will find the label and the Toolbar on the right end. If the label is absent the Toolbar is on the left.

Below are all the actions available in the Toolbar.

- **Add**: Add a row after the selected row. Added rows appear in light green color in the table. Updated rows will be shown in light orange color.
- **Insert**: An item selection flyout shows up. You can add rows to the Table from the items which are already available.



Components of item selection flyout will be as follows:

- **Search Options**: A search input field with a search button is available at the top within the flyout. By clicking on the search button, you filter the items grid.
- **Property Filter**: By clicking over the Property Filter check box available below the search, you perform advanced search operations on group containing various options to create custom expressions.
- **Items Grid**: All the items of that ItemType are shown in the grid with respect to the Search & Property Filter conditions.
- **Inserting Rows**: By clicking on the Ok button at the bottom right, you add all the items selected to the Table after the selected row or at the end. Added rows appear in light green color in the table.
- **Close**: By clicking on the Cancel button at the bottom right or the close button at the top right, you close the flyout without inserting.
- **Open Related Item**: By clicking on the Open Related Item action, you open a new tab by loading the related item of the selected row.
- **Open Relationship Item:** By clicking on the Open Relationship Item action, you open a new tab by loading the relationship item of the selected row.
- **Up**: By clicking on the Up action, you move the selected row above the previous row.
- **Down**: By clicking on the Down action, you move the selected row below the next row.
- **Delete**: By clicking on the Delete action, you delete the selected row from the Table.
- **Lock**: By clicking on the Lock action, you lock the selected row for exclusive edit. If the associated item with row is locked by someone else, you won't be able to lock the row.
- **Export Excel**: By clicking on the Export Excel action, you download all the visible data shown in the table to an excel file.
- **Export PDF**: By clicking on the Export PDF action, you download all the visible data shown in the table to a PDF file.

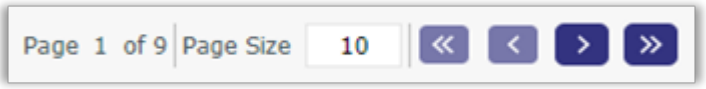
- **Filter**: By clicking on the Hidden dropdown, you select Simple option to show column filters on the table. You enter filter values on individual columns and click on the search icon beside the hidden dropdown to filter the rows based on the criteria. Beside the search icon there is also a clear option to clear all the filters
- **Custom Action Buttons**: Additionally, based on the requirements admins can configure custom action buttons for the table.

These two actions below are **exclusively** available in the **Toolbar for the Dashboards**:

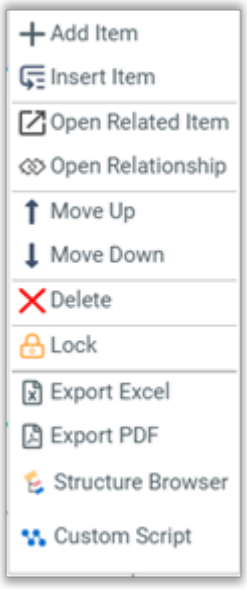
- **Property Filter**: By clicking over the Property Filter check box available below the search, you perform advanced search operations on group containing various options to create custom expressions. For more information regarding Property Filter, you can refer to section **Error! Reference source not found.** of this document.
- **Save**: By clicking over the save action button, save all the changes performed related to that table.

Below are other functionalities which are available for the table:

- **Sort**: Sort the table rows in ascending or descending order with respect to that column.
- **Drag & Drop**: Move and reorder the rows with ease. This feature is only available when the table is in scroll mode and reordering is enabled.
- **Column Filter**: By clicking over the filter icon in the column header, property filter containing various options to create custom expressions shows up automatically selecting property name and default first operator. You can enter filter values on individual columns and select search icon to filter the rows based on RelationshipType or ReferenceType properties. For table with RelationshipType properties, only “AND” logical operator is supported. Once property filter is applied, filter icon turns solid for the column on which filter is applied.
- **Pagination**: Pagination controls are shown at the bottom to navigate to other pages of the table. You can enter Page Size and select the search icon from the top action Toolbar to adjust the number of rows per page.



- **Context Menu**: By right clicking on any of the table rows, context menu shows up. All the Toolbar actions excluding the simple filter actions can also be accessed through the Context Menu.



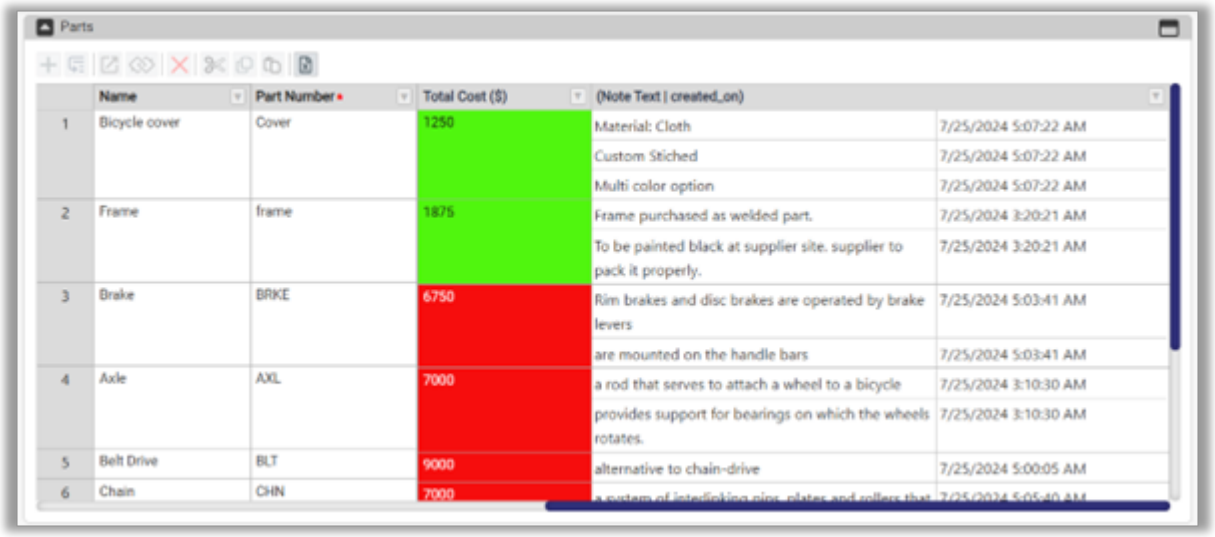
- **Multi Select**: You can also use CONTROL & SHIFT buttons to select multiple rows based on the requirement.

If Table columns have a fixed width, the following features are enabled:

- **Freeze Columns**: The columns which are frozen will be visible to access at the left of the Table, while you scroll horizontally to other columns.
- **Column Resizing**: Based on requirement, you can increase or decrease the width of the columns. Columns can be resized by dragging the right edge of the column header.
- **Column Re-Ordering**: Drag-drop columns to re-order them based on the requirement.

Worksheet

Worksheet control provides Excel experience for creating datasheet. You use Worksheet control to show relationship data or reference data of the context item as datasheet. When you add worksheet control to the form, properties of Relationship and Related ItemType are mapped to worksheet columns. Columns in the Worksheet are constrained with property data types. It supports nested rows from child relationships. Worksheet control has multiple settings that control its appearance and behavior at runtime.



Above the worksheet control, on the left end you will find the label and the Toolbar on the right end. If the label is absent the Toolbar is on the left.

Below are all the actions available in the Toolbar:

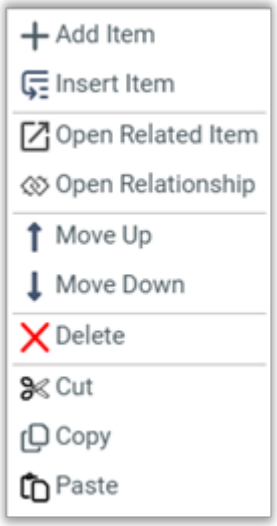
- **Add**: Add a new row after the selected row.
- **Insert**: Item selection flyout shows up. You can add rows to the Worksheet from the items which are already available.
- **Open Related Item**: Open a new tab by loading the related item of the selected row.
- **Open Relationship Item:** Open a new tab by loading the relationship item of the selected row.
- **Up**: Move the selected row above the previous row.
- **Down**: Move the selected row below the next row.
- **Delete**: Delete the selected row.
- **Cut**: Cut the selected cell or row.
- **Copy**: Copy the selected cells or rows to the clipboard.
- **Paste**: Paste the selected cells or rows from the clipboard.
- **Export**: Download all the data shown in the worksheet to an excel file.
- **Custom Action Buttons**: Admins can configure custom action buttons for the Worksheet.

These two actions below are **exclusively** available in the **Toolbar for the Dashboards**:

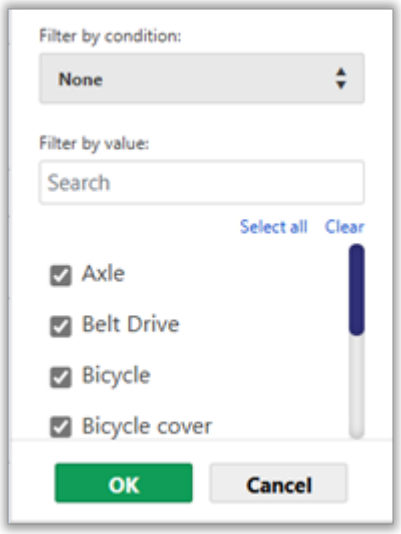
- **Property Filter**: By clicking over the Property Filter check box available below the search, you perform advanced search operations on group containing various options to create custom expressions.
- **Save**: Save all the changes performed related to that Worksheet.

Below are other functionalities which are available for the Worksheet:

- **Context Menu**: By right clicking anywhere on the worksheet, context menu shows up. You access Toolbar actions excluding the Export action through the Context Menu.

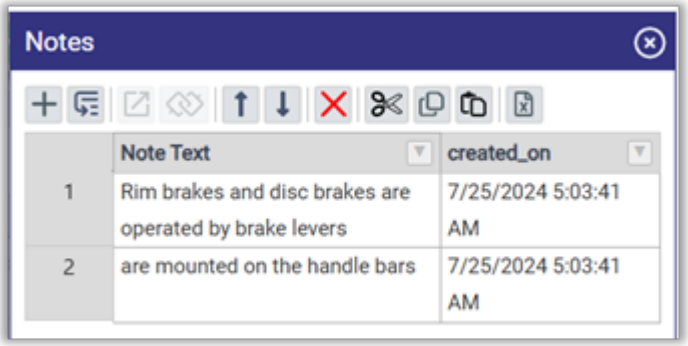


- **Sort**: Sort the rows in ascending or descending order with respect to that column.
- **Drag & Drop**: Select a row or multiple rows and use the drag & drop to move and reorder the rows with ease.
- **Multi Select**: Use CONTROL & SHIFT buttons to select multiple rows or cells based on the requirement.
- **Filter**: Available at the column level; access it by clicking on the arrow down icon in the column header to filter the rows with ease.



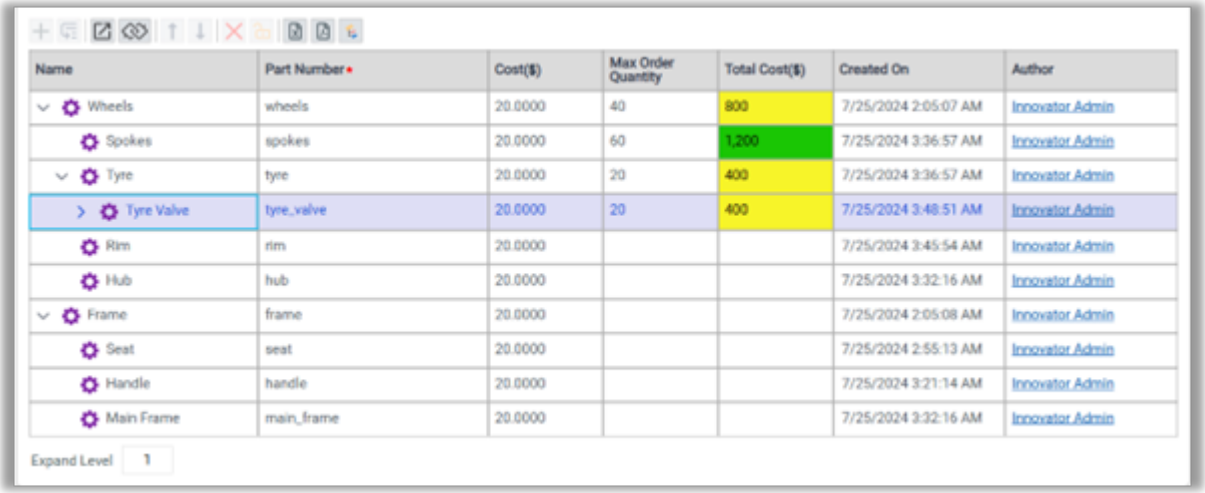
If the Worksheet columns have a fixed width, the following features are enabled:

- **Freeze Columns**: Frozen columns will be visible to access at the left of the Worksheet, while you scroll horizontally to other columns.
- **Column Resizing**: Increase or decrease the width of the columns. Columns can be resized by dragging the right edge of the column header.
- **Column Re-Ordering**: Drag-drop columns to re-order them based on the requirement.
- **Nested Rows**: Double click to edit nested rows data from the flyout.



TreeTable

TreeTable control shows the structure of items of the same ItemType. At the time of adding tree-table control to the form, properties of Relationship and Related Item Type are mapped to tree-table columns. TreeTable control has multiple settings that control its appearance and behavior at runtime.



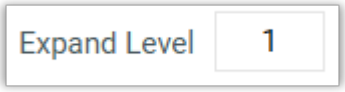
Above the TreeTable control, on the left you will find the label and the Toolbar on the right. If the label is absent the Toolbar is on the left.

Below are all the actions available in the Toolbar:

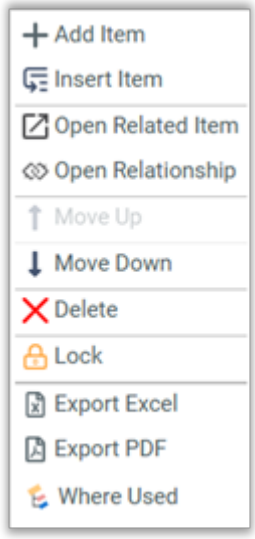
- **Add**: Add an item as a child or a sibling of the selected node.
- **Insert**: Item selection flyout shows up. Add rows to the TreeTable from the items which are already available.
- **Open Related Item**: Open a new tab by loading the related item of the selected row.
- **Open Relationship Item:** Open a new tab by loading the relationship item of the selected row.
- **Up**: Move the selected row above the previous row. Up action will be enabled only when the parent item is locked.
- **Down**: Move the selected row below the next row. Down action will be enabled only when the parent item is locked.
- **Delete**: Delete the selected row from the TreeTable. Delete action will be enabled only when the item is locked.
- **Lock**: Lock the selected row for exclusive edit. If the associated item with the row is locked by someone else, you won't be able to lock the row.
- **Export Excel**: Download all the visible data shown in the TreeTable to an excel file.
- **Export PDF**: Download all the visible data shown in the TreeTable to a PDF file.
- **Custom Action Buttons**: Admins can configure custom action buttons for the TreeTable.

Below are other functionalities which are available for the TreeTable:

- **Expand Level**: When you expand the node, it only gets its first level children. If you want to expand to multiple levels, the Expand Level value can be set. Expand Level is available at the bottom left of the TreeTable.



- **Context Menu**: Right click on the rows of TreeTable to see Context Menu; access Toolbar actions.



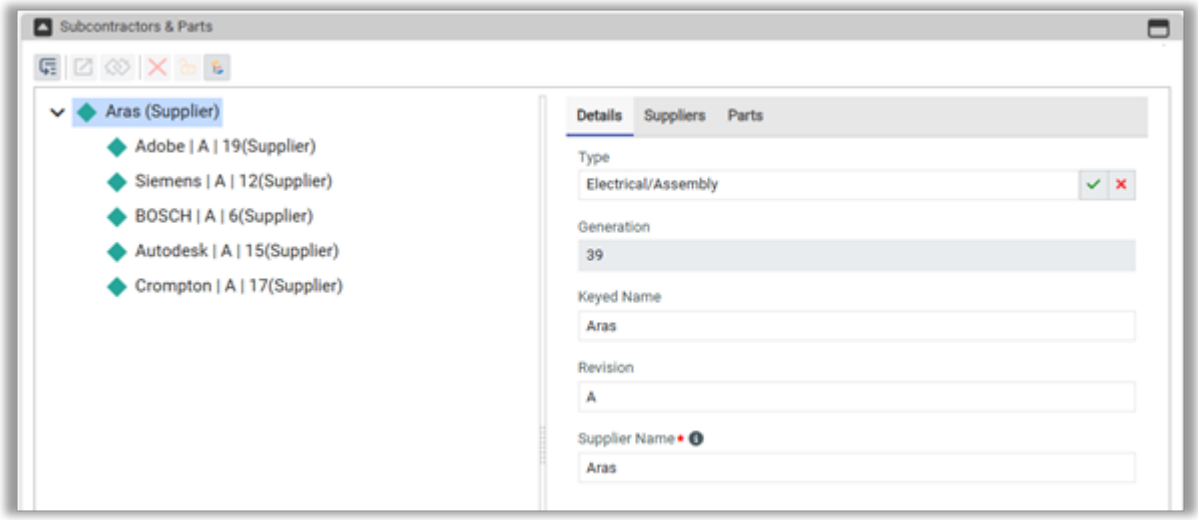
If the TreeTable columns have a fixed width, the following features are enabled:

- **Freeze Columns**: Frozen columns will be visible to access at the left of the Worksheet, while you scroll horizontally to other columns.
- **Column Resizing**: Increase or decrease the width of the columns. Columns can be resized by dragging the right edge of the column header.

Structure

Use Structure control to show any structure with items of different ItemTypes and Relationship Types:

- It is defined on Query Definition which requires context item Id. Use it as StructureBrowser and WhereUsed controls for a given ItemType by defining appropriate Query Definition.
- In edit mode, you can modify and save fields returned from the Query Definition back to Innovator.
- In edit mode, you can insert children of any allowed type at any level in the structure and save back to Innovator.
- You can configure Structure control to show relationships in the structure as nodes in the tree or rows in the table based on the configuration.



Above the Structure control, at the left you will find the label and the Toolbar on the right. If the label is absent the Toolbar is on the left.

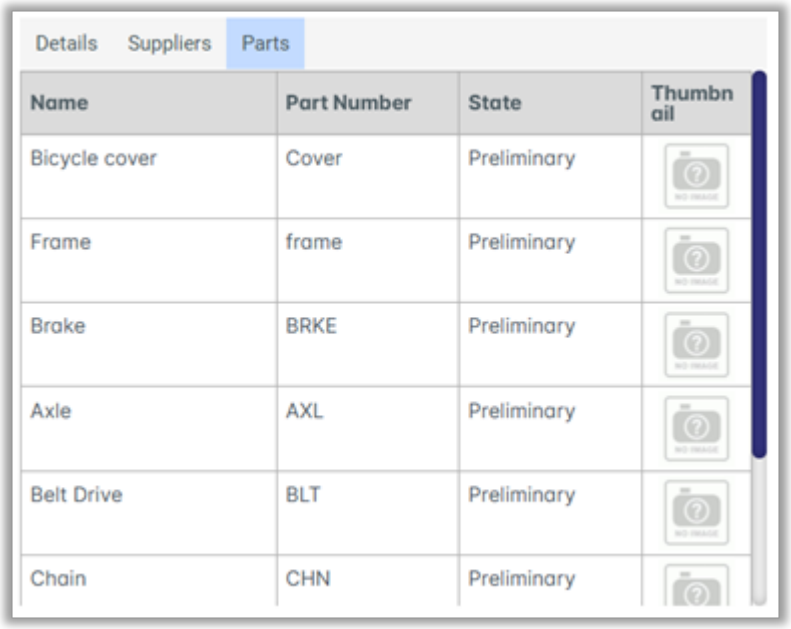
Below are all the actions available in the Toolbar:

- **Insert**: Item selection flyout shows up. Add rows to the Structure from the items which are already available.

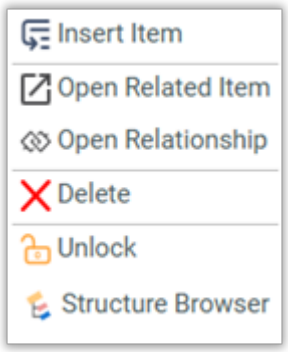
- **Open Related Item**: Open a new tab will be opened by loading the related item of the selected row.
- **Open Relationship Item:** Open a new tab will be opened by loading the relationship item of the selected row.
- **Delete**: Delete the selected node. Delete action will be enabled only when the item is locked.
- **Lock**: Lock the selected node for exclusive edit. If someone locks the associated item of the node, you won't be able to lock the row.
- **Custom Action Buttons**: Admins can configure custom action buttons for the Structure.

Other functionalities which are available for the TreeTable:

- **Separator**: Structure control is divided into two sections by the Separator. On the left, you have a Tree View of nodes and on the right there is a Details view of the selected node. By dragging the Separator, you can adjust width on both sections.
- **Tree View**: A classic tree view, with the current item as root node and all its relationships, references & related items configured are shown as its child nodes.



- **Details View**: Shows all the tabs configured. Details tab shows all the configured properties of the selected node item. Other tabs show relationships, references and related ItemTypes ("*" next to the tab name).
- **Sort**: Sort rows in ascending & descending order.
- **Context Menu**: Right click on the nodes of Structure to see Context Menu; access Toolbar actions.

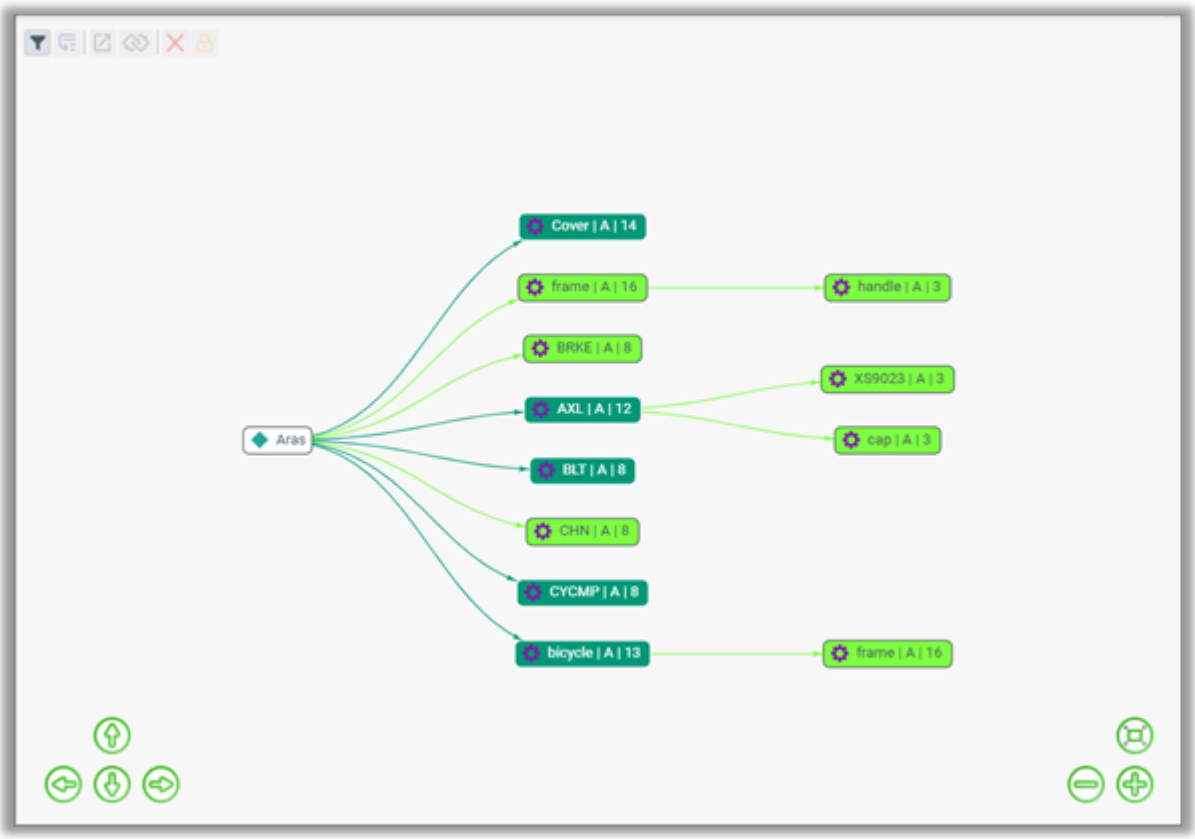


Graph

Use Graph control to show structure data of the context item in the graphical structure. Graph control is bound to a Query Definition which defines which Relationship Types, Related Item Types and properties to show in the control. By double clicking any node in the Graph, a flyout will show with the selected item properties along with its relationships. Each node in the graph will be shown with different color based on the background color set at RelationshipType or conditional formatting defined for the RelationshipType in the template definition.

Graph control comes with two modes:

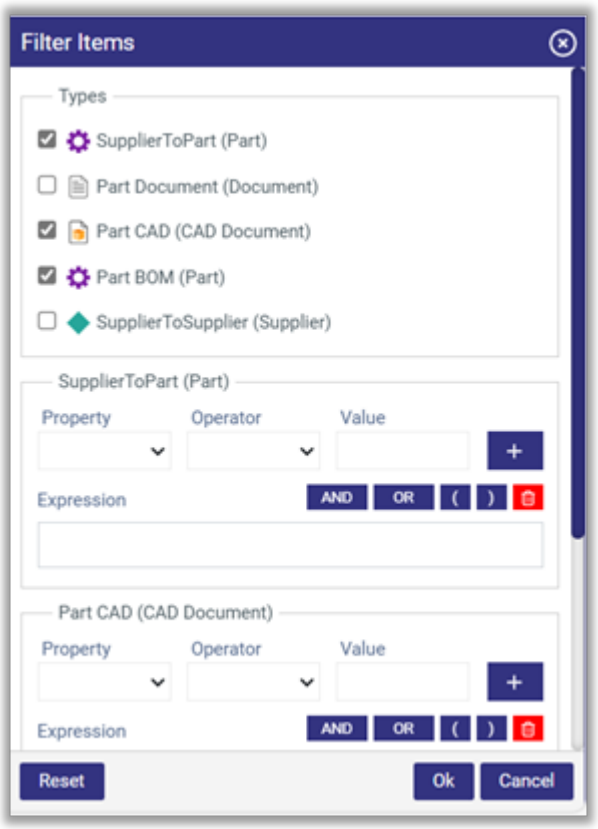
- **read-only** mode shows structure and properties, you can only navigate the structure and see properties of the selected node. You can also open the specific node in its own tab by selecting Open icon from the toolbar.
- in the **edit mode**, you can insert new nodes in the structure by selecting the Insert icon from the toolbar. In the InsertNode Flyout, allowed RelationshipTypes of the selected node are shown, by selecting specific RelationshipType and keyword. You can also edit properties or delete selected node.



Above the Graph control, at the left you will find the label and the Toolbar on the right. If admin hides the label, the Toolbar is on the left.

Below are all the actions available in the Toolbar.

- **Item Filter**: A filter flyout shows up from which you can filter the items in the graph view. All the ItemTypes shown in the graph view are in the first group and for each selected ItemType, property filter is shown in its individual group.



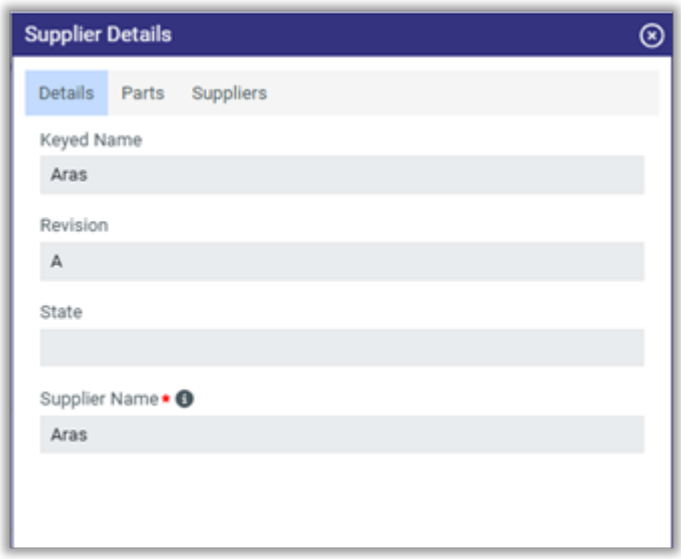
  - Filter flyouts layout will be as follows:
    - **Reset**: Reset all the filter selections to default.
    - **Ok**: Apply selection filters over the Graph and close the flyout
    - **Cancel**: Close the flyout.
- **Insert:** An item selection flyout appears. You can add nodes to the Graph from the items which are already available. All functionality and features are similar to those in the Table.
- **Open Related Item**: Open a new tab by loading the related item of the selected node.
- **Open Relationship Item:** Open a new tab by loading the relationship item of the selected node.
- **Delete**: Delete the selected node. Delete action will be enabled only when the item is locked.
- **Lock**: Lock the selected node for exclusive edit. If someone locks the associated item of the node, you will not be able to lock the row.
- **Custom Action Buttons**: Admins can configure custom action buttons for the Graph.

Other functionalities of the Graph are as following:

- **Graph Buttons**:
  There are arrow buttons available at the bottom left of the graph that are used to move the graph in their respective directions.

There are three other buttons available at the bottom right of the graph, those are Zoom In, Zoom Out & Fit to Size.

- **Node Details Flyout**: By double clicking on the node of the graph, Node Details Flyout shows up. Shows all the tabs configured. Details tab shows all the configured properties of the selected node item. Other tabs show relationships, references and related ItemTypes (“*” next to the tab name).

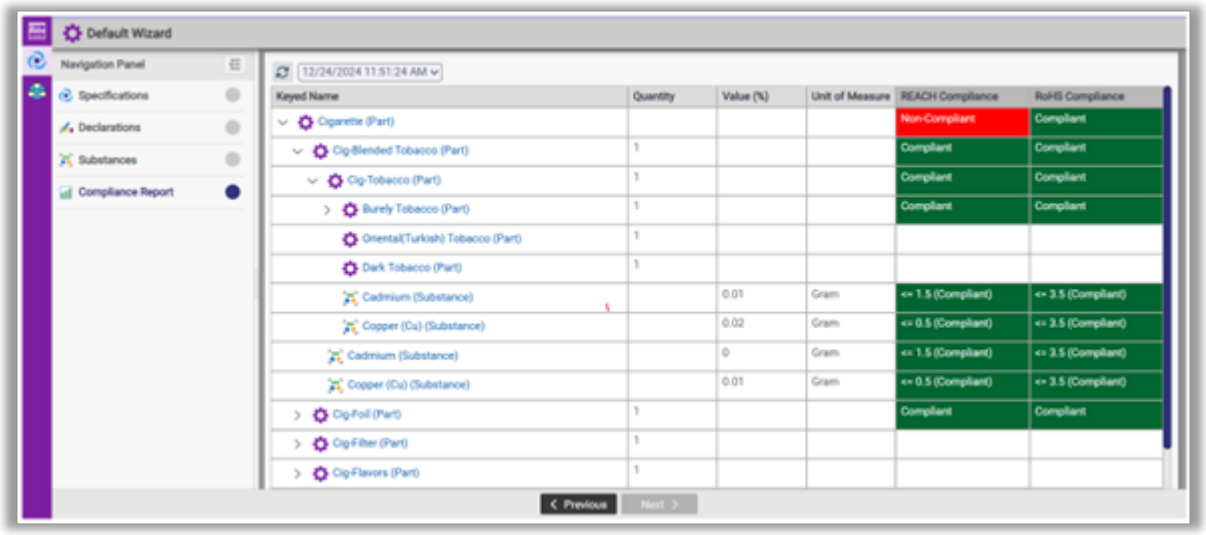


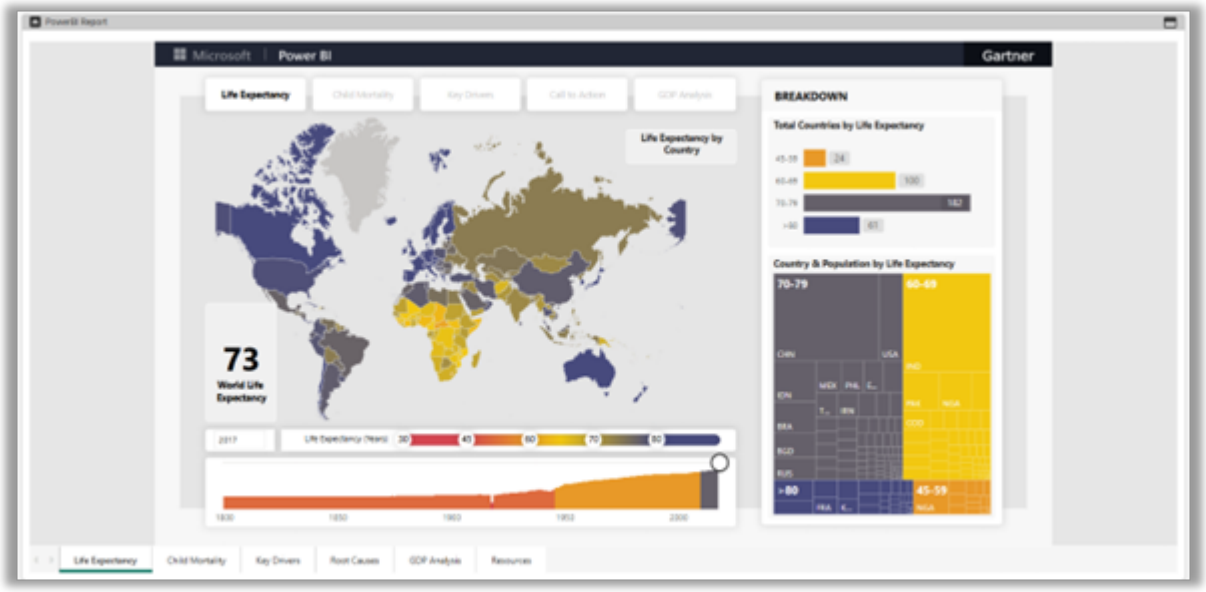
- **Sort**: Sort the rows in ascending and descending order.
- **Ok**: Apply all changes made in the flyout to the Graph and close the flyout.
- **Cancel**: Close the flyout.

Report

Use Report control to generate reports from the available data. Data is fetched for a report control based on the Query Definition selected for the control. Report is generated based on the data associated with the context item. It also can be generated based on global data of any other ItemType. Report control shows the data in the form of PivotTable, Line Chart, Bar Chart, Area Chart TreeTable, PowerBI. Report control is displayed on Item Details and Dashboard pages.

Type	State	Preliminary	Released	Totals
Assembly		182	36	**218**
Component		97	197	**294**
Material		2		**2**
Phantom		9		**9**
Software		14		**14**
Totals		**304**	**233**	**537**



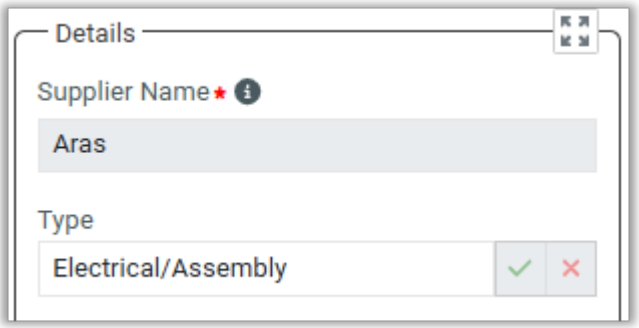


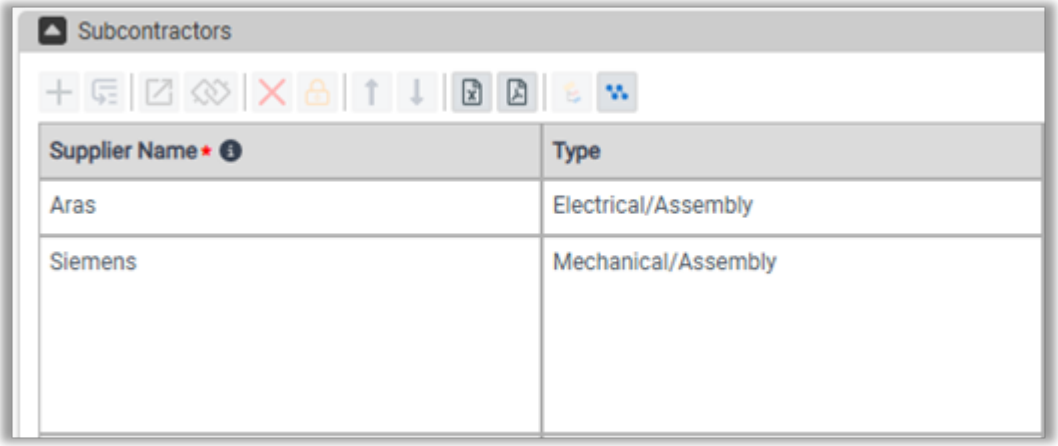
Drilldown Report is displayed by double-clicking on the row or column data for each subcategory. Individual item data can be opened by clicking hyperlink of the item from the Flyout.

Page & Control Features

Mandatory Properties

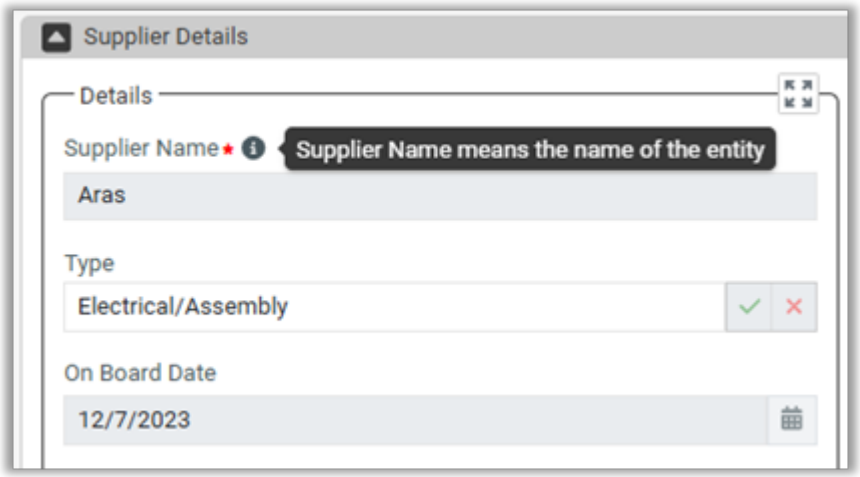
Mandatory properties on the form and collection controls are displayed with ‘*’ after property label.

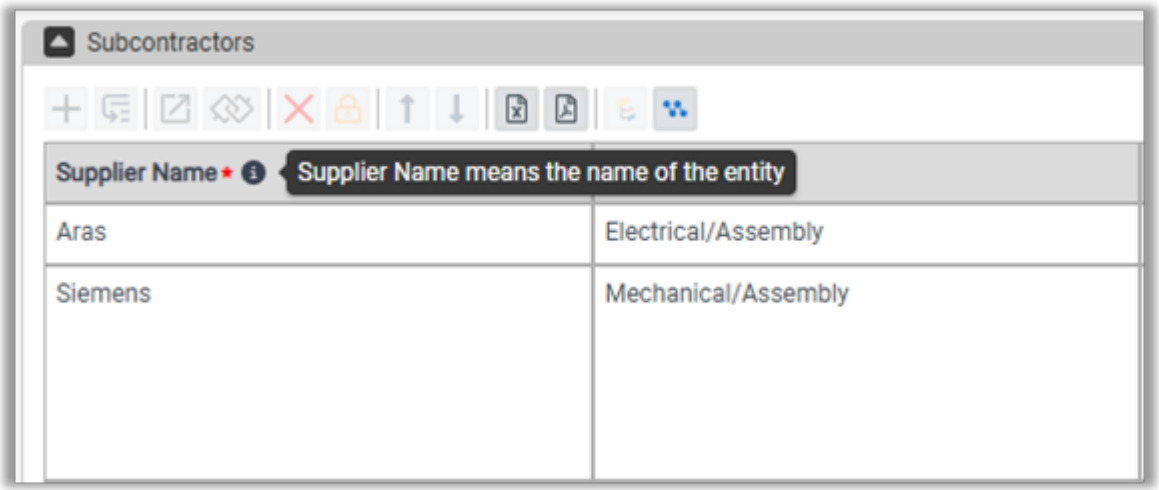




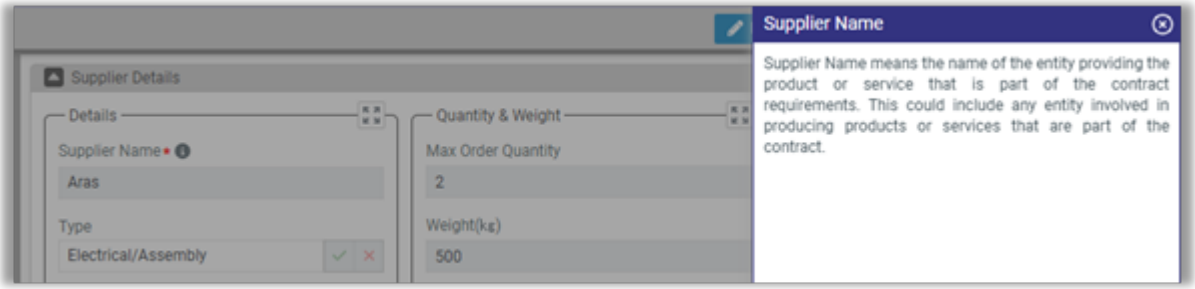
Tooltip & Help Text

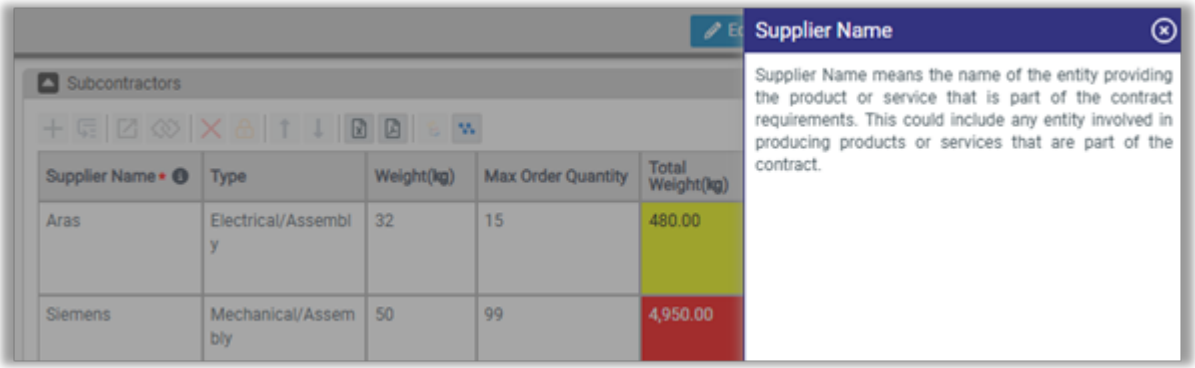
Tooltip is displayed when you hover the mouse on the information icon on the form field labels or collection control column headers if admin has setup the tooltip on the ItemType property definition.





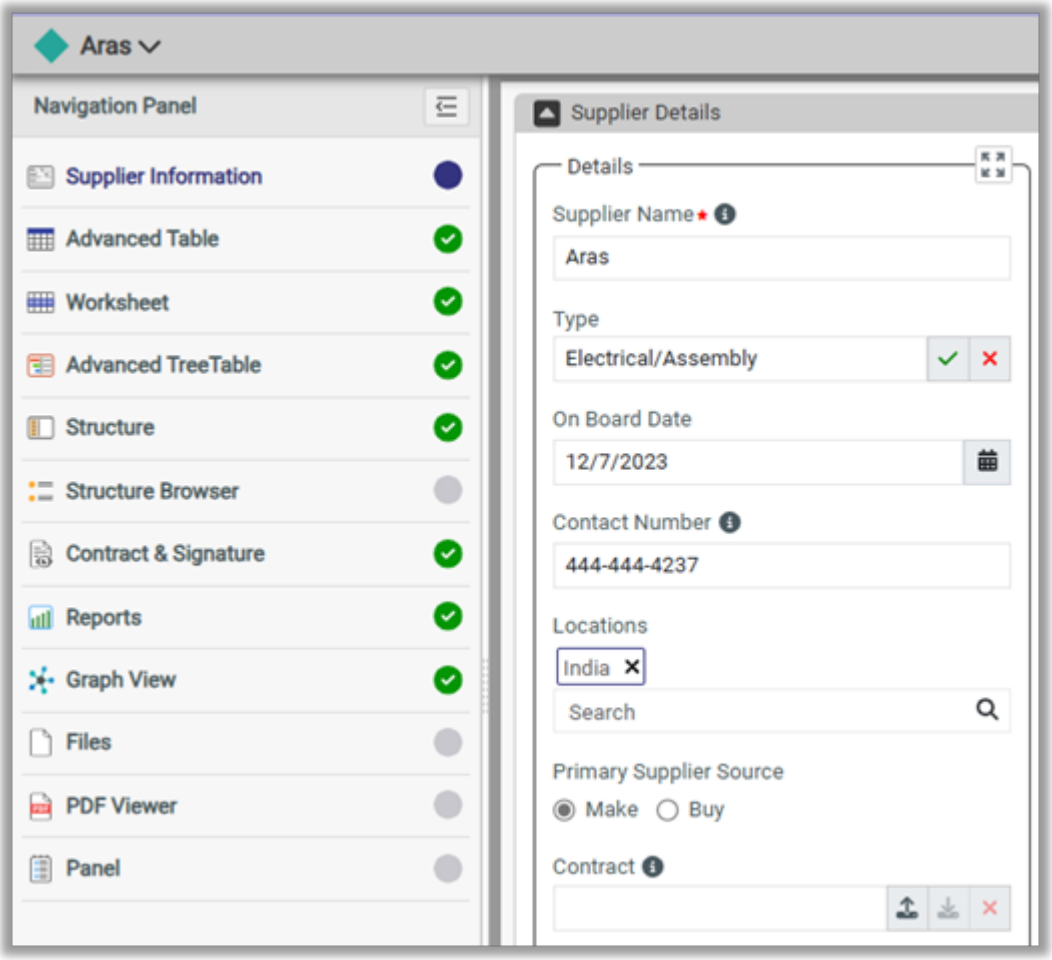
Help Text flyout will be shown when you click on the information icon on the form field labels or collection control column headers if admin has setup the help text on the ItemType property definition.





Display Conditions

Display Conditions on a control are defined with expression to dynamically render or hide controls based on property values. Display conditions defined on the control will decide whether control should be shown or hidden from the page. Display conditions can be defined on the pages as well as all control types. In the following example, ‘Contract & Signature’ page is visible only when ‘Show Subcontractors’ is selected.

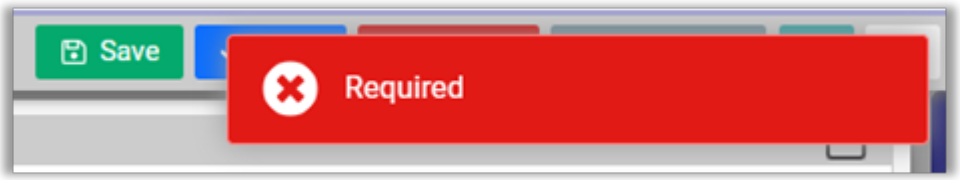


Validation Rules

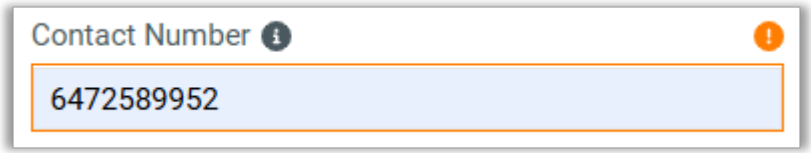
Validation Rules are defined on a control to validate entered data before submitted request to server:

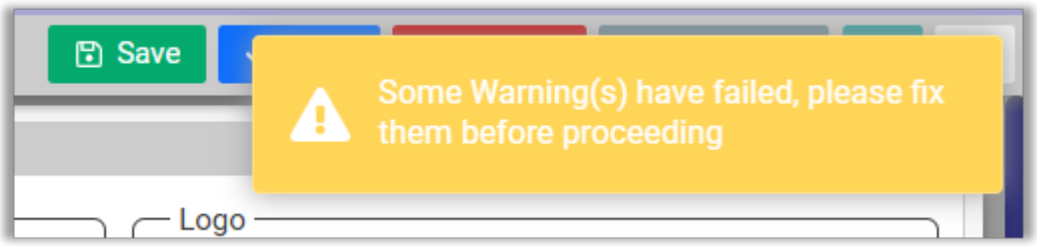
- Rules can be built based on the expression from the properties of the current and previous pages. As part of moving away from the page, all validation rules on the page are evaluated.
- If any of the rules are not satisfied, it shows popup dialog with Validations Failed message and icons against each control.
- Hovering the mouse on the icon to display specific errors or warning messages based on the defined validation rules on the control.
- You can move away from the current page by clicking any other page from the Navigation Panel.
- If the current page has any validation errors, a popup dialog will display with options to ignore and move forward or fix the issues before moving forward.
- If multiple rules are defined on the controls, all rules should be satisfied in order to move forward.

Validation can be of **Error** or **Warning** type. When validation error rule fails, error will be displayed in the popup dialog with options to ignore and move forward or fix the issues before moving forward. If you move forward without fixing validation errors for that page, in the navigation panel, page title with be shown with red-circle with tick icon to indicate there are validation errors in the page.



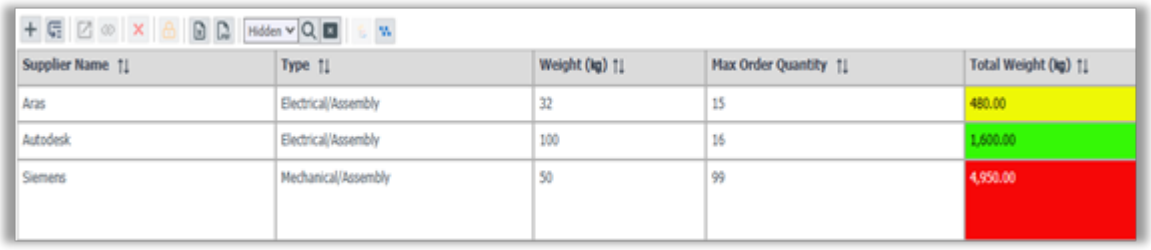
When validation warning rule fails, warning will be shown in the flash notification with orange background color. Warnings can be ignored, by clicking on Save, Done, Next buttons second time to move forward with the warning





Computed Values

Computed values are defined on the Text control to show calculated value when control is rendered. Computed value can be calculated based on the expression, expression can refer other properties from the current page or previous pages. If Text control is defined with computed value and it is marked as read-only, it can dynamically change its value based on the values entered in other controls on the page. In the following example, ‘Total Weight’ is calculated based on the values set for ‘Max Order Quantity’ and ‘Weight’ at real-time.



Supplier Name	Type	Weight (kg)	Max Order Quantity	Total Weight (kg)
Aras	Electrical/Assembly	32	15	480.00
Autodesk	Electrical/Assembly	100	16	1,600.00
Siemens	Mechanical/Assembly	50	99	4,950.00

Conditional Formatting

Conditional Formatting can be defined on Text control and Table control column to highlight the fields or cells with defined styles. Rules are built with expression based on thresholds set on property value. In the following example, Conditional Formatting is set on ‘Total Weight’ to highlight control with red color if the value is greater than 2000.

Dynamic ReadOnly

Dynamic ReadOnly allows any control to be marked as read-only at runtime even if you can edit the property associated with that control with lock. Dynamic read-only is a Boolean expression defined using properties from the current and previous pages. At runtime, Dynamic expression will be evaluated and control will be marked as read-only if the value is true.

Reset Dependencies

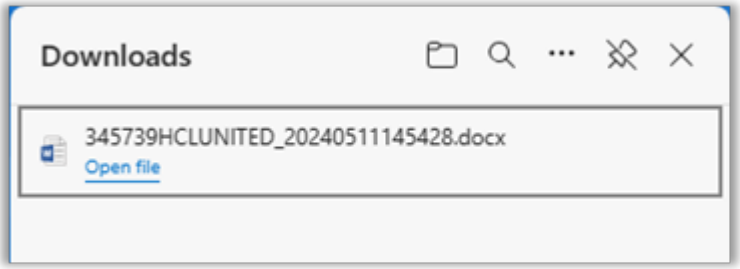
Reset dependencies can be defined on a control, to reset its value when any of the controls defined in reset dependencies have been changed at runtime.

Data Export

Word/PDF Generation

Automates creation of MS Word and PDF documents based on properties from the item:

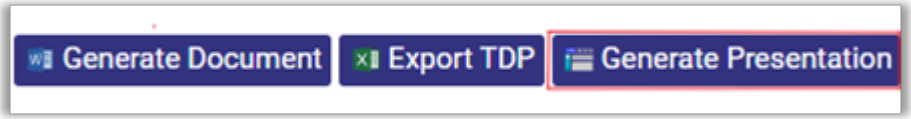
- Word document is created based on the Word template defined in the system.
- Word template has content controls mapped to properties of ItemTypes and RelationshipTypes.
- Can be defined with chart, data-bound with Query Definition to dynamically generate chart with data in the generated document.
- Document can be composed from the documents stored in Innovator.
- Can have a repeating content controls with DataSource generated from relationships collection or ServerMethod output.
- Document generation with watermarking and PDF can be configured based on lifecycle states.

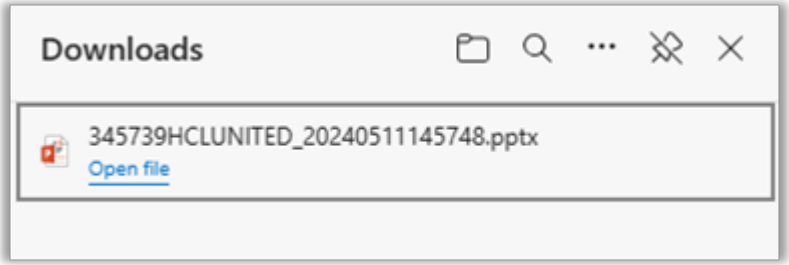


PowerPoint Generation

Automates creation of MS PowerPoint document based on the properties from the item:

- PowerPoint document is created based on the PowerPoint template defined in the system.
- PowerPoint template has content controls mapped to properties of ItemTypes and Relationship Types.
- Can be defined with chart, data-bound with Query Definition to dynamically generate chart with data in the generated document.
- Can be composed from the PowerPoint documents stored in Innovator.
- Template can have table and dynamic list controls with DataSource generated from relationships collection or ServerMethod output.

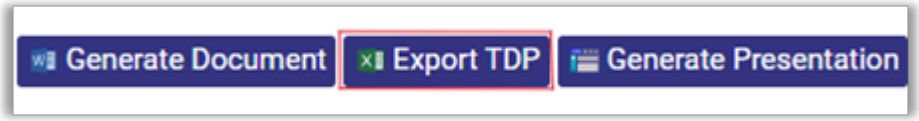


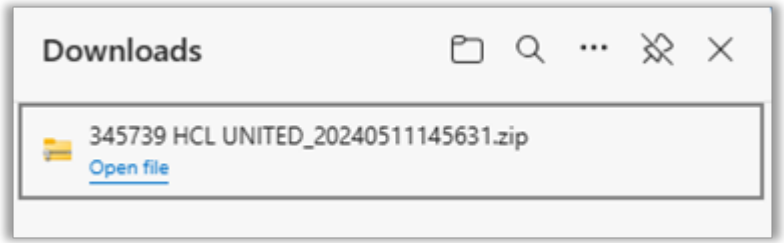


Technical Data Package

TDP provides offline access to outside suppliers for specific data and files:

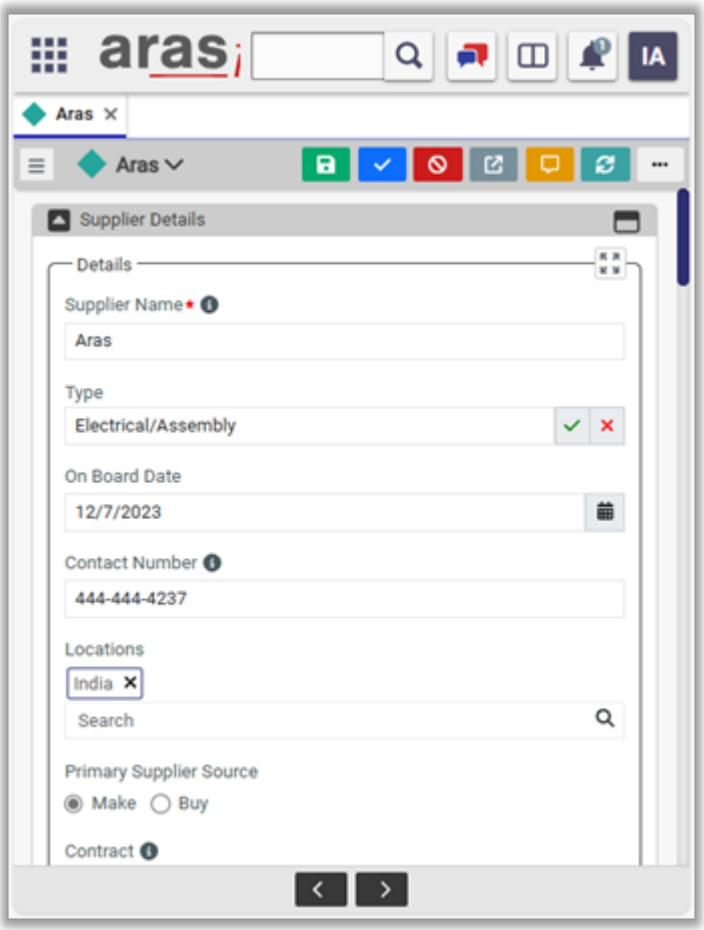
- TDP contains items data, relationships data and files bundled in a zip file.
- TDP is created based on configured excel template with multiple worksheets, having columns with properties from ItemTypes, Relationship Types, and Files.
- Package is generated on the server based on user action, lifecycle promotion, or scheduled job.
- Generated package has an excel file with all items and relationships data along with the files.

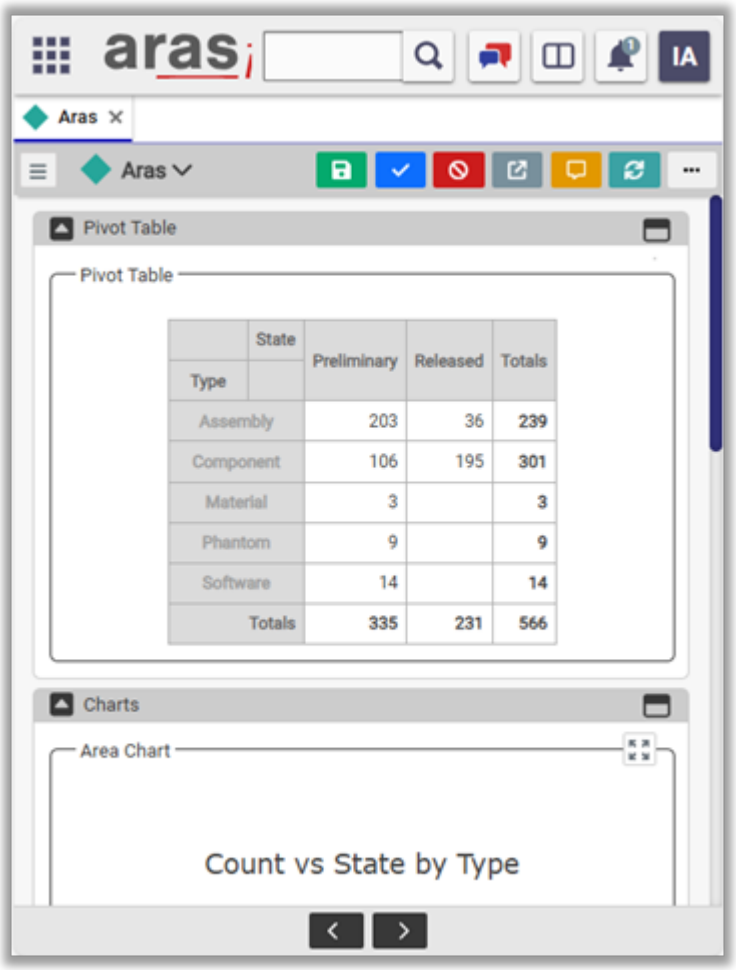




Responsive UI & Mobile Views

Controls rendered on the page are adjusted to the available page width to avoid showing horizontal scrollbars. In the mobile device, all the pages are shown with a mobile layout.





Aras Presentation Generator - User Guide

Introduction

Purpose

This User Guide describes how to use the Aras Presentation Generator for external users.

Scope

This document provides instructions for configurations and functionality of the Aras Presentation Generator application.

Target Audience

This document is intended for external users with limited access to data.

Guide Organization

This document describes how to use the Aras Presentation Generator application. This User Guide is organized into the following sections:

SECTION 1: **Introduction**	This section defines the purpose and scope of the User Guide and identifies its intended audience. It also provides a concise summary of each part to assist users in navigating the guide effectively.
SECTION 2: **Introduction to Presentation Generator**	This section explains the purpose of the Presentation Generator and provides an overview of the features available in the Aras Presentation Generator. It also includes a description of the template creation process.

Introduction to Presentation Generator

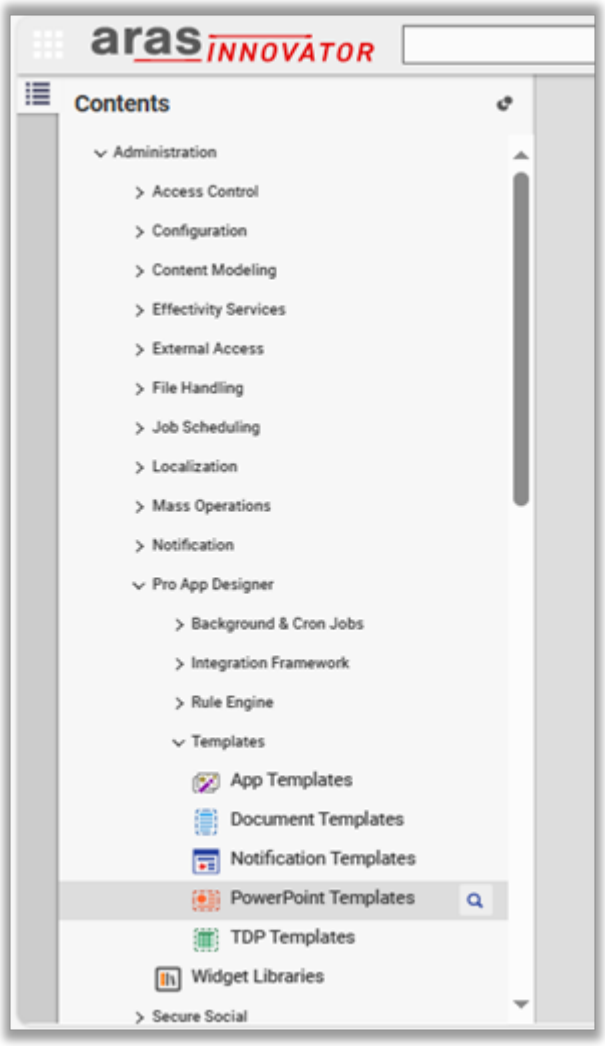
The Presentation Generator automates the creation of presentations based on user selections made through wizard pages. Presentations are generated using a configured template and do not require any plugins or Office connectors. Generation occurs on the server and can be triggered by user actions, lifecycle promotions, or scheduled jobs.

Templates include content controls mapped to properties of ItemTypes and RelationshipTypes. Presentations can also be composed from existing presentations stored in Innovator. Templates may contain repeating content controls, which are linked to data sources derived from relationship collections or ServerMethod outputs. The appropriate template for generation is selected based on predefined rules.

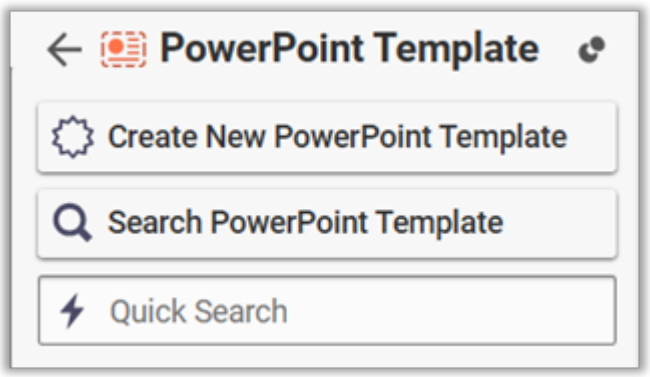
Presentation generation can be initiated by any user action or system event. The process utilizes a template associated with the relevant ItemType.

Presentation Generator Template Configurations

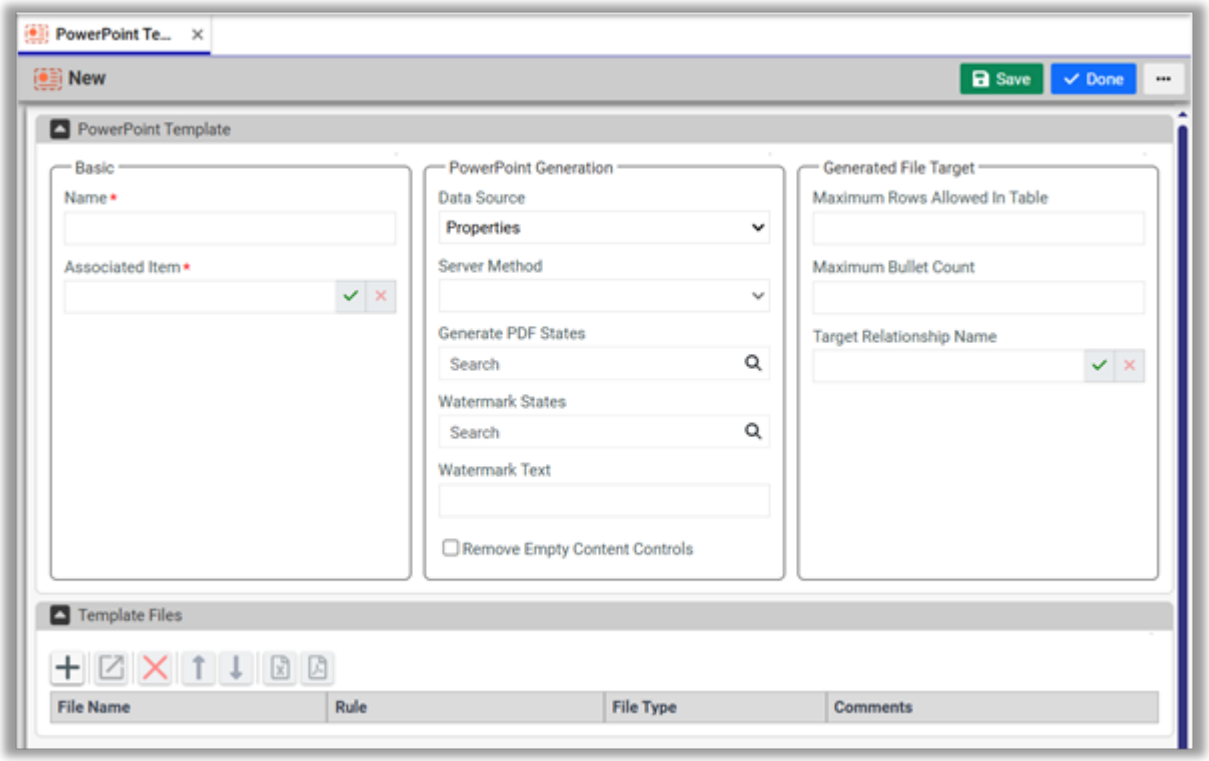
1. Go to TOC à Administration à Pro App Designer à Templates à PowerPoint Templates.



2. Click on Create New PowerPoint template.



3. Enter the necessary information, save your work, and add the template file accordingly.



Following are the details for PowerPoint Template properties:

Basic:

- **Name**: The name of the PowerPoint template is a required property and must be specified in order to save the template.
- **Associated Item**: The context item for which the presentation template is being created is a required property and must be specified to save the template.

PowerPoint Generation:

- **Data Source**: There are two options available for selection under Data Source. The first option is 'Properties' (selected by default), and the second option is 'Server Method'.
- **Server Method**: Server method selection is available when 'Server Method' is chosen as the data source. All methods configured as Server Events on the associated item will be listed for selection.
- **Generate PDF States**: A list of states for which the PDF will be generated must be specified.
- **Watermark States**: Specifies the list of states during which watermarks should be applied when generating the PDF.
- **Watermark Text**: Specifies the text to be used for watermarking. This attribute is applicable only when the 'Watermark States' attribute is defined.
- **Remove Empty Content Control**: Determines whether empty content controls should be removed from the generated PDF. This setting is applied during the presentation generation process.

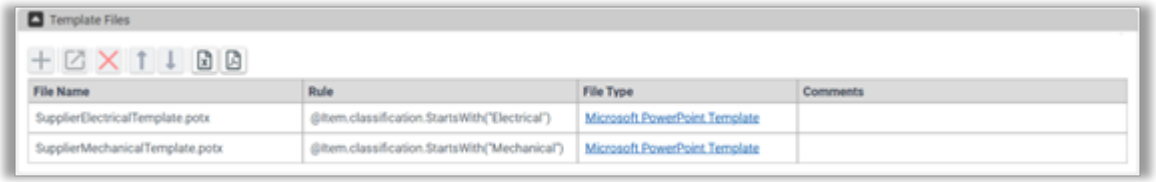
Generated File Target:

- **Maximum Rows Allowed in Table**: Specifies the maximum number of table rows that should be displayed on a single slide.
- **Maximum Bullet Count**: Specifies the maximum number of bullet points that should be displayed on a single slide.
- **Target Relationship Name**: Specifies the name of the relationship to which the files will be attached after processing.

Template Selection

Each Presentation Generator template can have multiple Presentation files attached. During the Presentation creation process, the appropriate PowerPoint document is automatically selected based on predefined rules:

- If a presentation file does not have any rule defined, it will be selected by default. In cases where multiple files lack rules, the first available presentation file will be chosen automatically.
- If all presentation files have rules defined, the first file that successfully passes validation will be selected. If none of the files meet the validation criteria, the presentation generation process will be aborted.



Presentation Template Creation

Presentation generation application uses following content controls:

Plain text Control

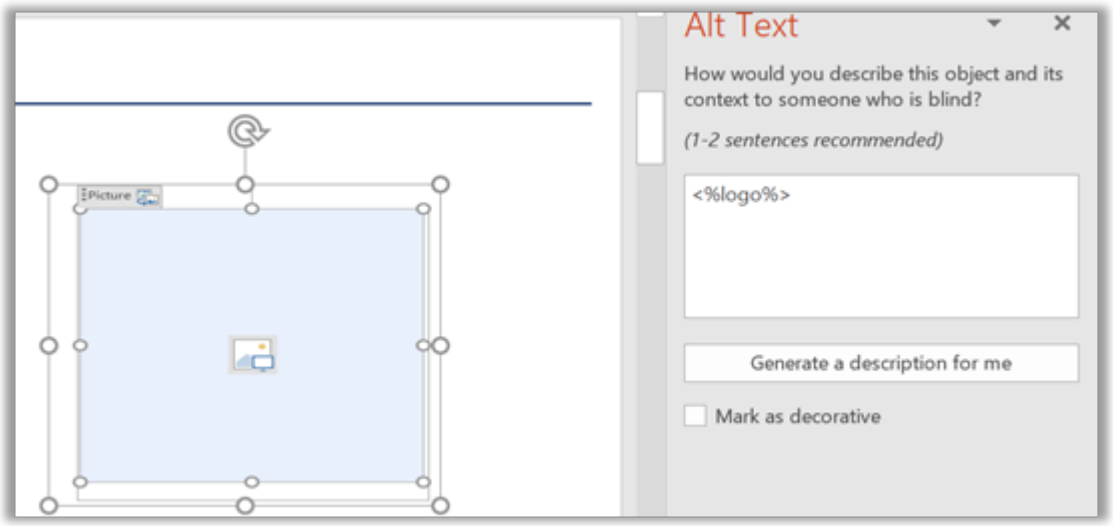
This is used to display plain text.

Examples:

- **Contact Number**: <%contact_number%>
- **Supplier Products**: <%SupplierToProduct%>
- **Locations**: <%locations%>

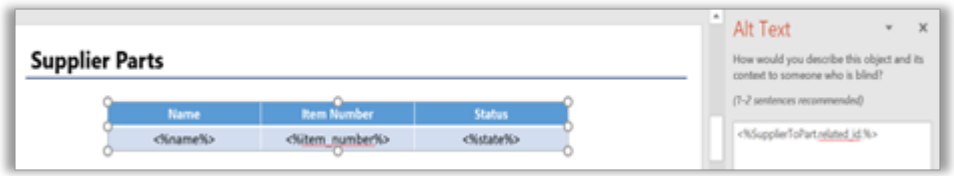
Image

The 'Alt Text' property of an image object is used to define the mapping to an Innovator Item property. This mapping enables the system to retrieve the image from the specified property during presentation generation.



Repeating Text Content: Tabular Presentation

Repeating content controls are used to populate tables in the presentation. The 'Alt Text' property of the table object defines the related path, which is used to retrieve related items. Each column in the table is then mapped to a property of the related item.



Repeating Text Content: Per Slide

The 'Alt Text' property of the slide title is used to define the path to related items. Properties of these related items can be referenced within the content of the slide. Slides will be repeated dynamically for each related item retrieved.

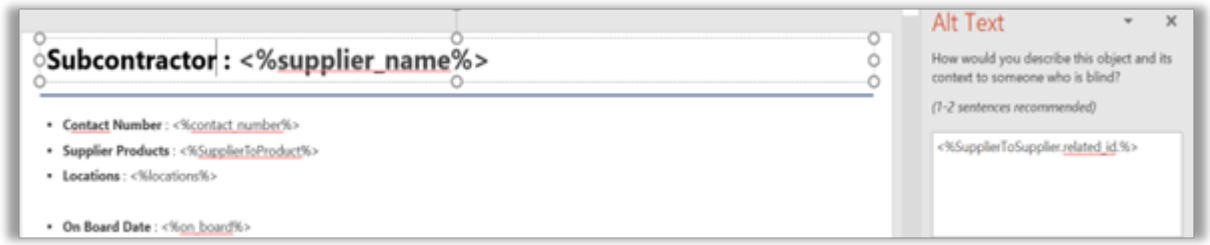
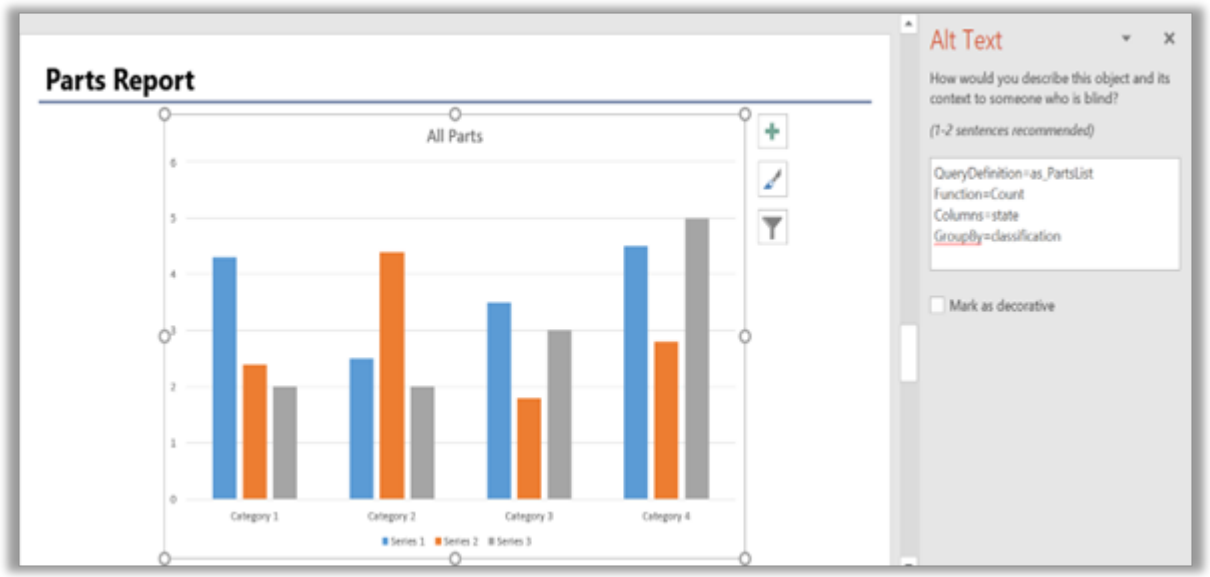


Chart Control

Query definitions are used to retrieve data for chart controls. All chart configurations are specified within the 'Alt Text' property of the chart object. Please refer to the image below.



Tag Naming Conventions and its Meaning

In the above image tag value is given as <%supplier_name%>. Let us see what it means

1. **<%..%>** Indicates, processing at server end. Please note that it is mandatory to follow this convention.
2. **<%supplier_name%>** In this context, ‘supplier_name’ refers to the name of a property within the Supplier ItemType. By using this tag, the application identifies the corresponding property value in Aras Innovator and substitutes it into the presentation at the designated placeholder.
3. **<%SupplierToPart.related_id.name%>** Here involves relationship.
    1. ‘SupplierToPart’: is the RelationshipType of Supplier Item Type.
    2. ‘related_id’ is the property pointing to Part Item Type.
    3. ‘name’: is the property of Part Item Type.

So, this tag value actually fetches the value of Property ‘name’ attached to ‘Part’ item type.

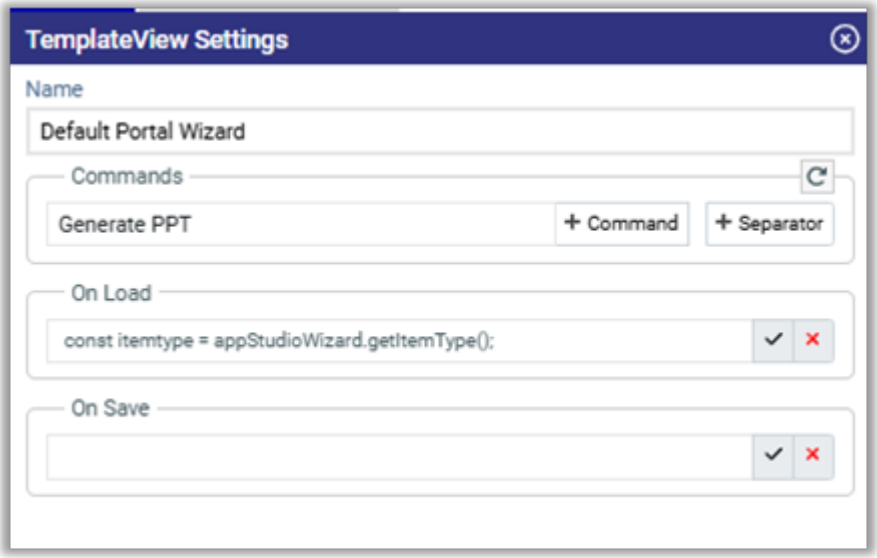
Using this naming convention, the server follows a specific sequence when processing the tag:

1. Go to ‘SupplierToPart’.
2. Search for related_id and get the Data Source value which is ‘Part’ in this case.
3. Go to Part and access the ‘name’ property value.

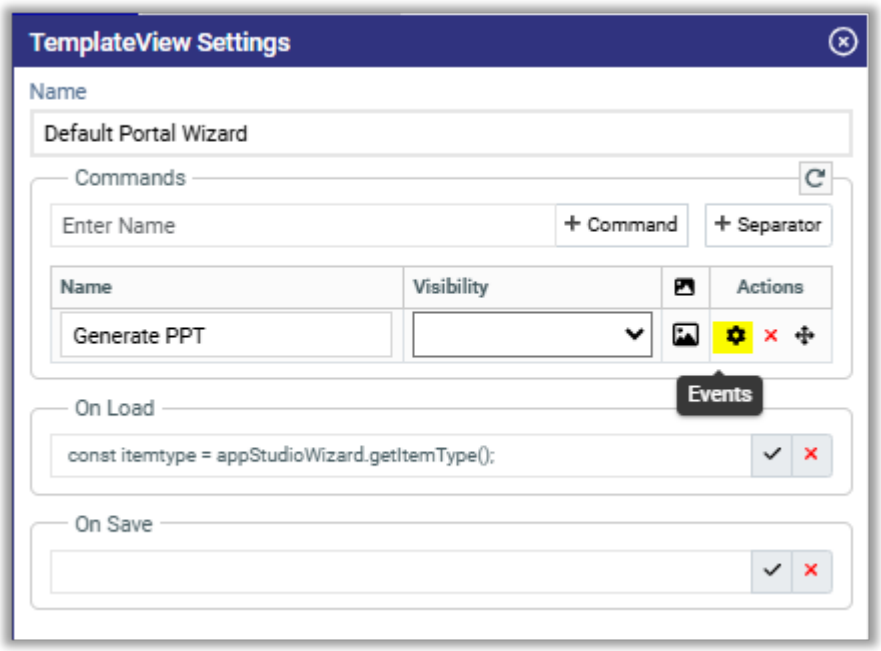
Configure Presentation Generator Command on Template View

To enable presentation generation using this generator, it can be configured as a command on the ItemType template by following these steps:

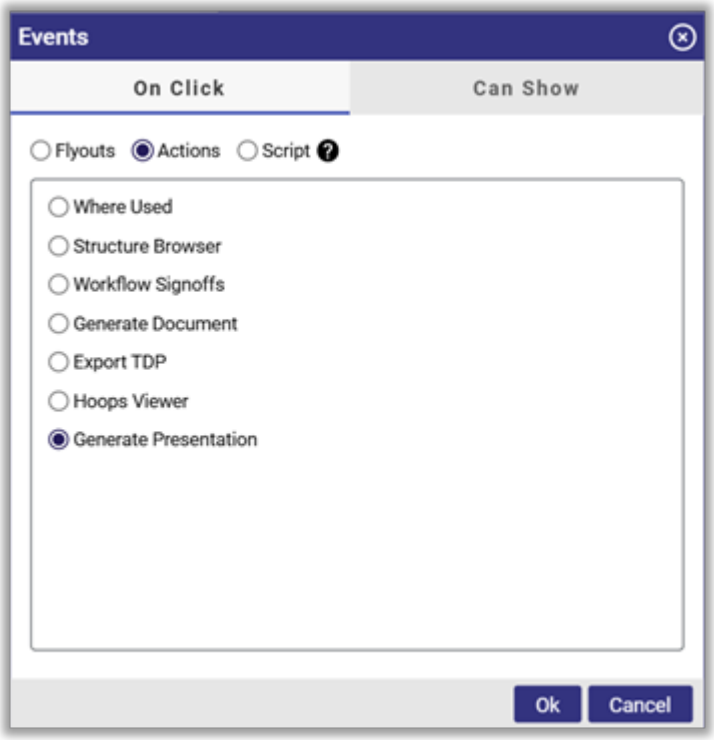
1. Go to TOC → Administration → Pro App Designer → Templates → App Templates →Open App Template and then Open Template View.
2. Go to More → Settings → Add Command name (like below image) → Click on Command button.



3. Click on Events (highlighted in below image).



4. Select Actions and then Select Generate Presentation, click Ok.

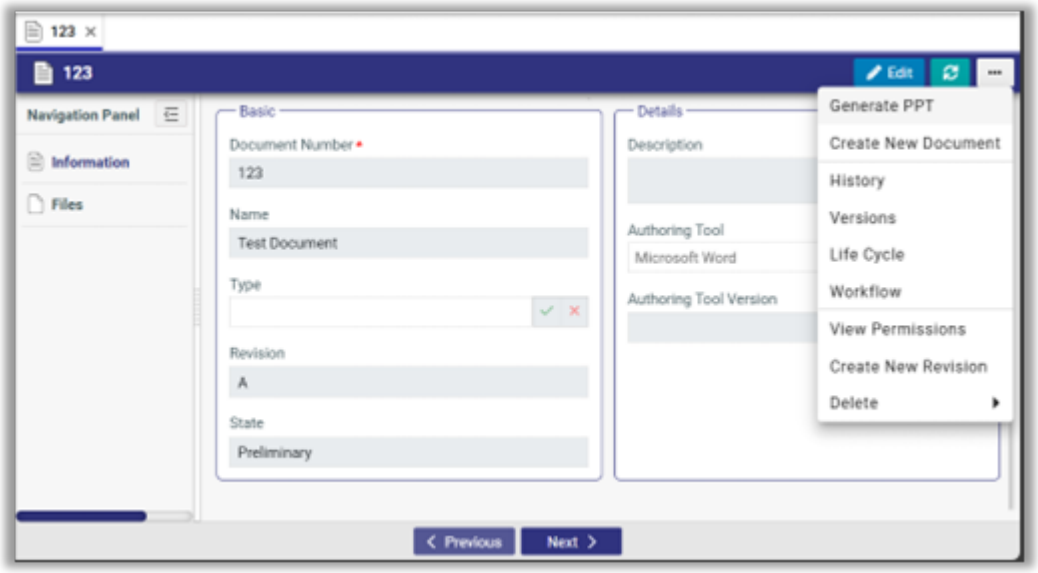


5. Add Visibility then click on Ok

6. Example of Commands with different visibility

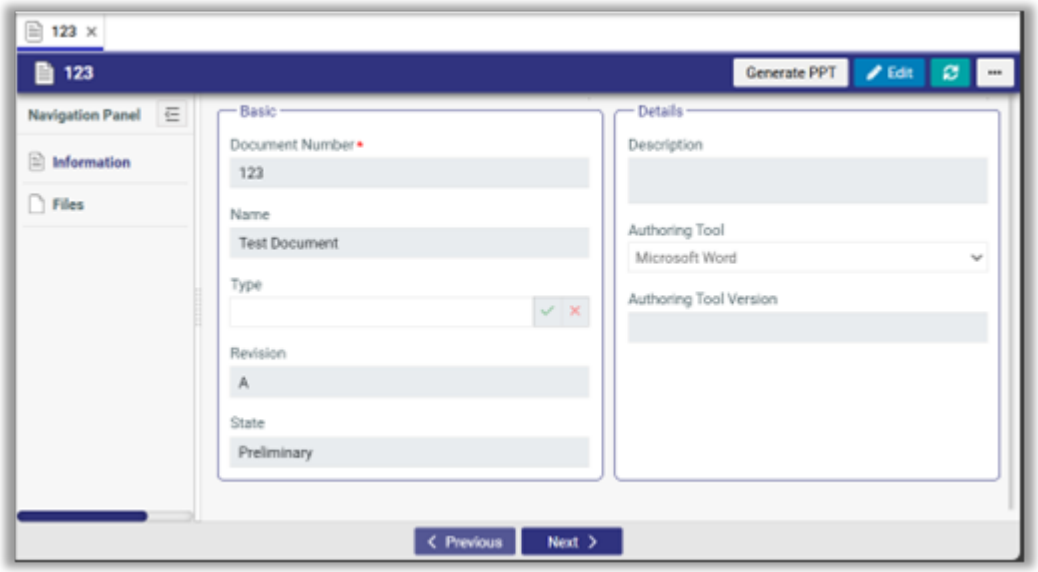
Visibility: Show In Dropdown

eg.



Visibility: Show In ButtonGroup

eg.

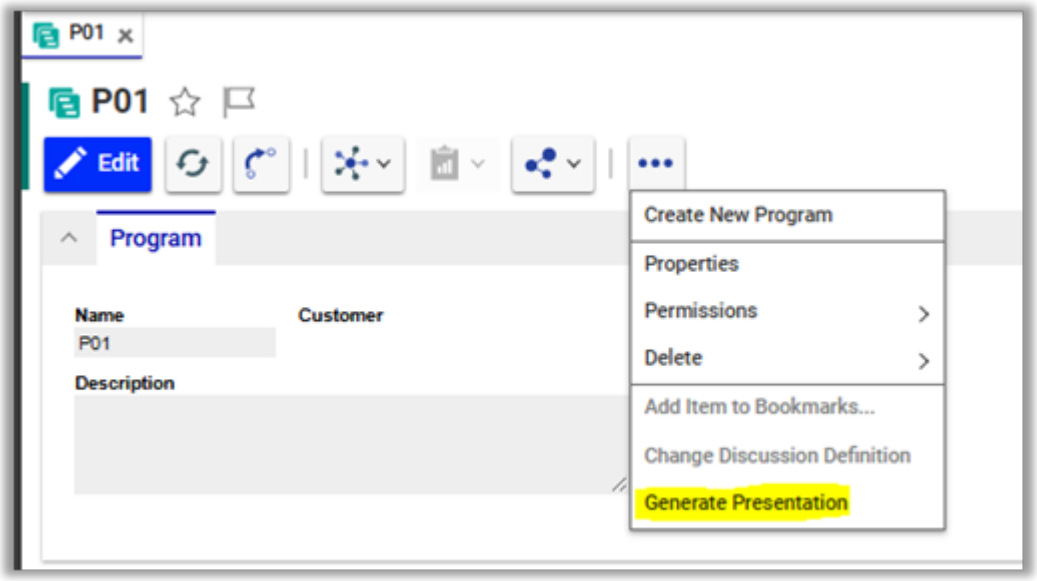


Configure Custom action for Presentation Generator

To generate presentations using the Presentation Generator, it can be configured within Aras Innovator as either custom actions or CUI (Client User Interface) menu items. Steps to use it through custom action:

1. Create a custom action on the context item type.
2. Create and attach a client method that invokes the server method named “as_GeneratePresentation”, which internally utilizes the PresentationService provided by the Presentation Generator.
3. The “as_GeneratePresentation” method can be executed on a context item, which is then used to replace property values within the presentation template.

   e.g.



4. Additionally, a custom server method can be introduced to modify the values of the context item before initiating presentation generation. Instead of directly calling the “as_GeneratePresentation” method, this custom method can include logic to set or transform property values on the context item. Once the necessary adjustments are made, a PresentationService object can be instantiated, and its APIs can be invoked.

   e.g.

```
 Item contextItem = inn.newItem(itemType, "get");
 contextItem.setID(itemId);
 contextItem = contextItem.apply();
```

```
 Prorigo.Plm.PresentationGenerator.IPresentationService
presentationService = new
Prorigo.Plm.PresentationGenerator.Aras.PresentationService(inn, CCO);
```

```
 var fileItem = inn.newItem("File");
 var fileId = presentationService.GeneratePresentation( contextItem,
"as GetDocumentAssociationJson", false ); //Boolean parameter to
indicate file to be checked in to item as per configuration or not
 fileItem.setID(fileId);
 var resultItem = fileItem.apply("get");
 return resultItem;
```

ArasProApp Designer - Installation Guide

Introduction

Purpose and Target Audience

This installation guide provides the required information for successfully installing and configuring Aras ProAppDesigner for Aras Innovator administrators. The installation involves updates to both the code tree and database.

Prerequisites

To install ProAppDesigner, **users need to confirm** that the installation of Aras Innovator meets the minimum requirements to run ProAppDesigner.

Check the eligibility of code tree and database:

From the **TOC**, select **Administration**, **Variables**.

VersionMajor	VersionMinor	VersionServiceUpdate
14	0	22
14	0	23
14	0	24
14	0	25
14	0	26

If it does not meet the service pack requirements listed in the previous steps, contact Aras to discuss options.

Make sure to have the **following versions** installed:

- Aras Update Tool **1.17+**
  or
- Aras Innovator **Package Import Export Utilities**

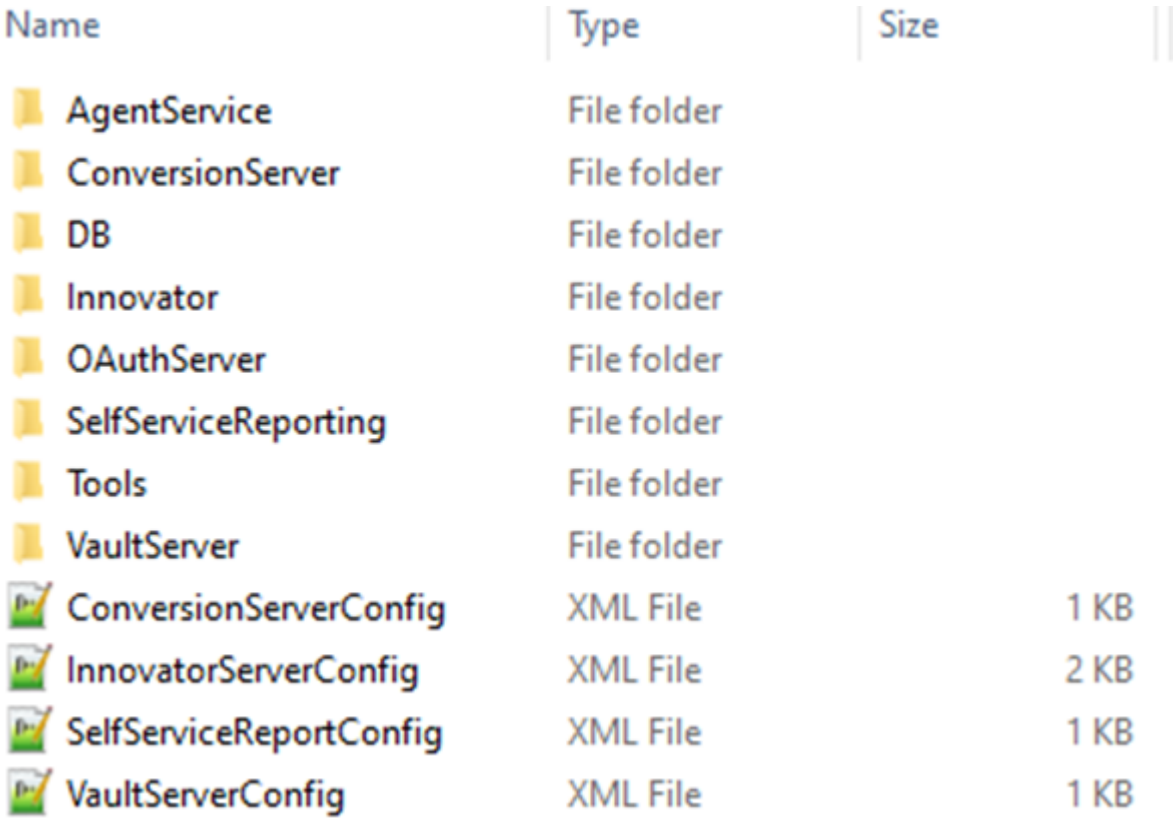
Installing ProAppDesigner

ProAppDesigner must modify the Aras Innovator database with the metadata required to run the application.

Notification and Backup

1. **Notify users** that the system will be down at a scheduled time, they should log out prior to that.

2. **Backup** the code tree.
   The **Code Tree** refers to files and folders installed to the disk when Aras Innovator was first installed.
   The default path for the Code Tree installation would be something like:
   ```
   C:\Program Files (x86)\Aras\Innovator
   ```
   With the following contents:

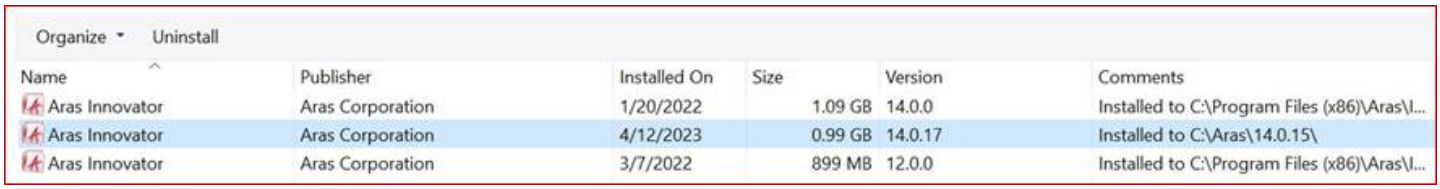


Follow these steps to confirm the installation folder:

1. From the Windows search field, enter **Control** to access the **Control Panel.**
2. Select **Programs and Features.**
3. Right-click the **Name** column and select **More…**
    1. Select the Comments header if it is not checked.

2. Click OK.

4. Search for the **Aras Innovator** entry in the **Programs and Features** window.
5. View the value set in the **Comments** column for the Aras Innovator entry:



After the tree code installation path has been verified, back up the folder and all its contents:

- **Disconnect** all users from the database.
  The easiest way to prevent client sessions from committing any further changes to the database is to change the database connection string in the InnovatorServerConfig.xml from <DB-Connection … to <xDB-Connection and restart the w3svc service (IIS). This expires all session and prevents all new connections to the Aras Innovator database through the existing instance.
- **Backup** the database.
- Place these files in a **safe location**, as they will be needed to restore if the process fails.

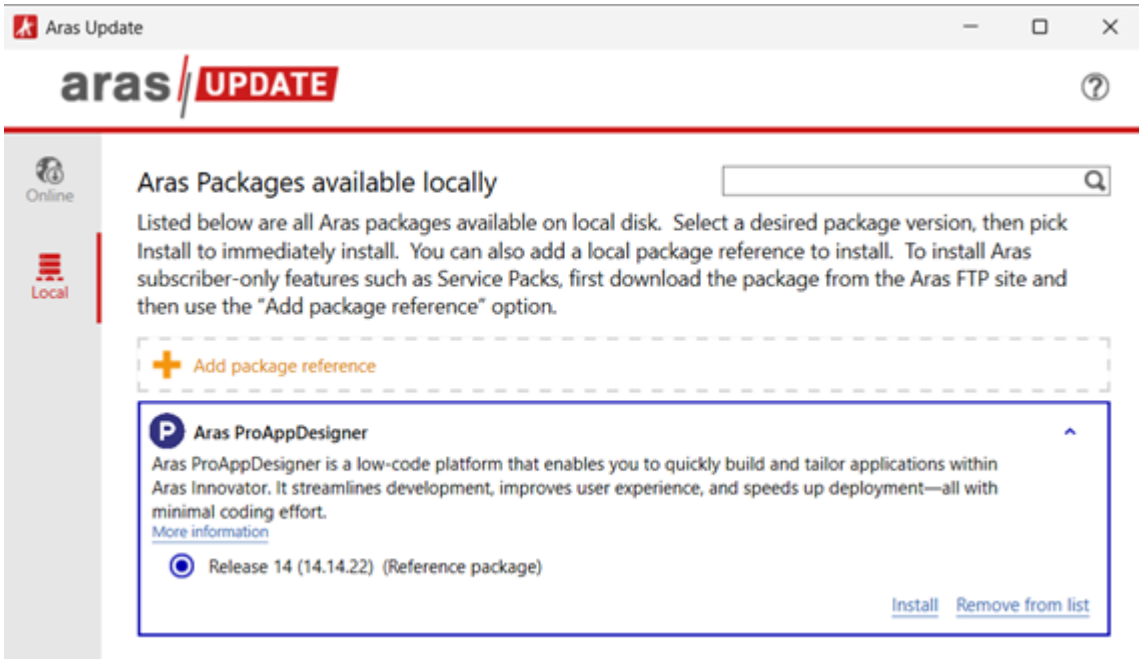
- **Enable** database connections.
  After the database backup has been completed, enable the database connection string in the InnovatorServerConfig.xml by changing it from <xDB-Connection … to <DB-Connection and restarting the w3svc service (IIS).

Installing the Aras ProAppDesigner

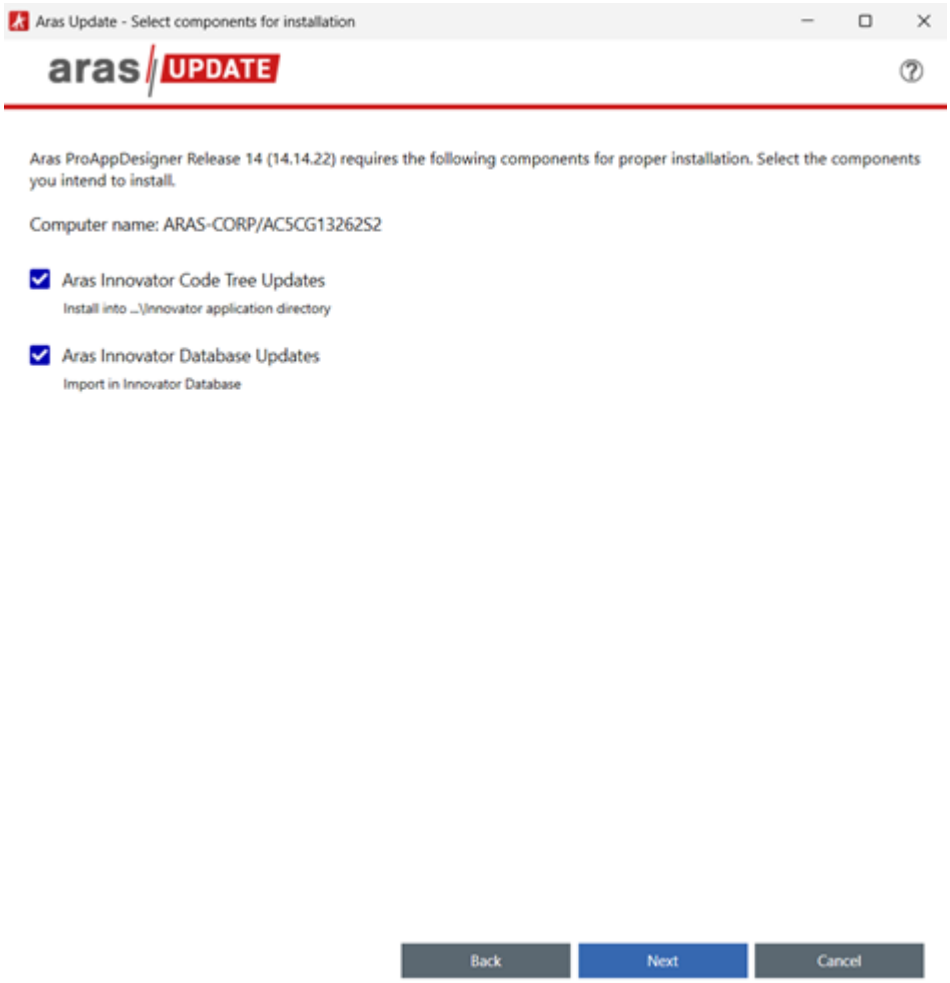
The ProAppDesigner can be installed using either the **Automated Installation** option (using the Aras Update tool), or the **Manual Installation** option (using the Aras Import Tool).

Automated Installation Option

1. Download the **Aras ProAppDesigner CD Image** from the Aras File Sharing site.
2. Unzip the Aras ProAppDesigner CD Image on the local computer.
3. Enable the **Super User** login.
4. Launch the **ArasUpdate.exe** file as administrator.
5. On a default installation, the ArasUpdate.exe file can be found in the C:\Program Files `(x86)\Aras\Aras Update` \ directory.
6. Locate and select the **Aras ProAppDesigner** package.

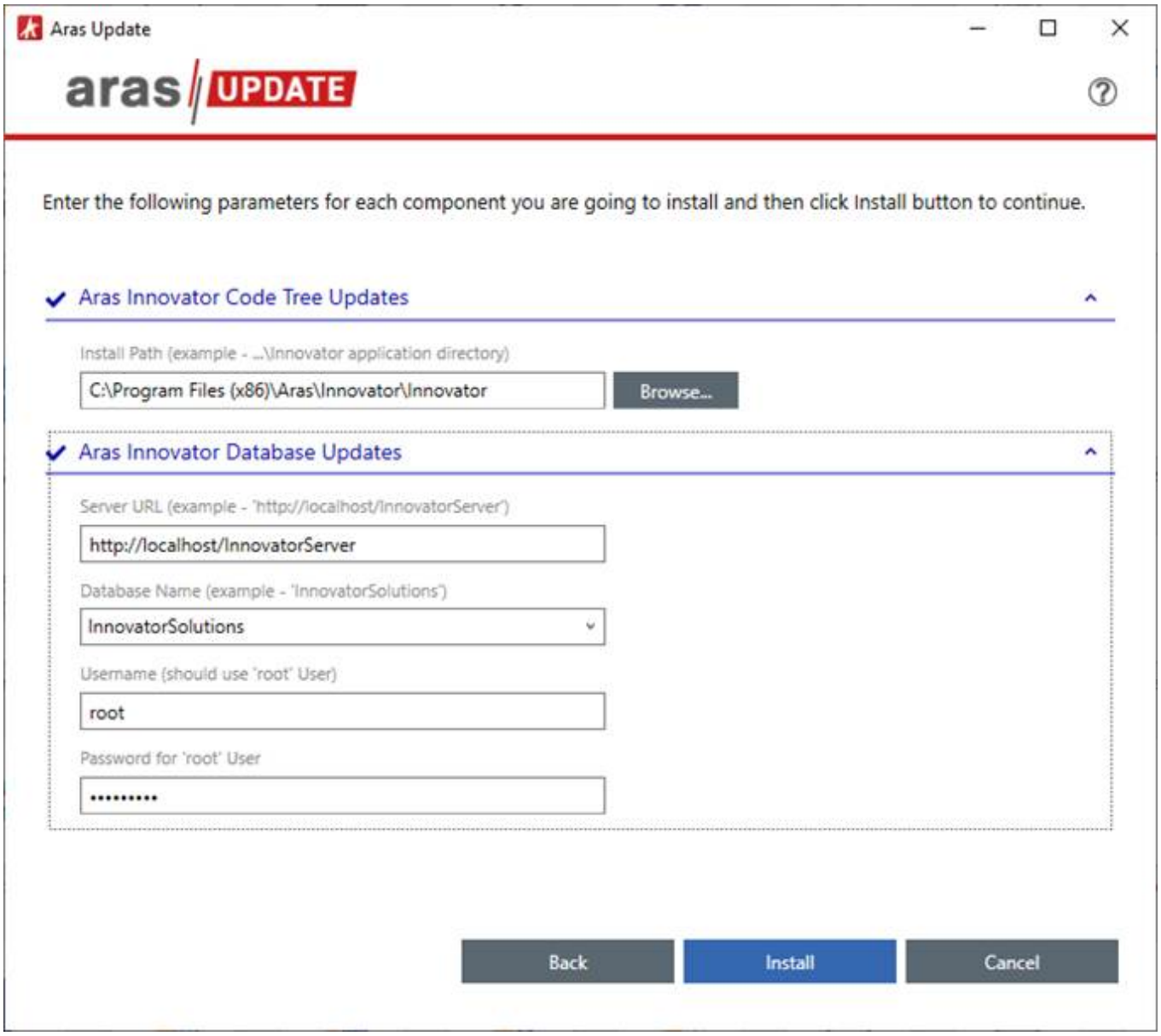


7. Click **Install**.



8. Select the **Aras Innovator Code Tree Updates** and **Aras Innovator Database Updates** options.
9. Click **Next**.
10. Select the appropriate logging option and click **Next**.
    The logging option is used to record the installation attempt and can be used to troubleshoot issues.
11. Input the connection information and click **Install**.
    - Server URL = The connection URL for Aras Innovator
    - Database = The target Aras Innovator database
    - Username = root
    - Password = Password for “root” login (Default is “innovator”)

Aras Update should import everything required for Aras Innovator ProAppDesigner.

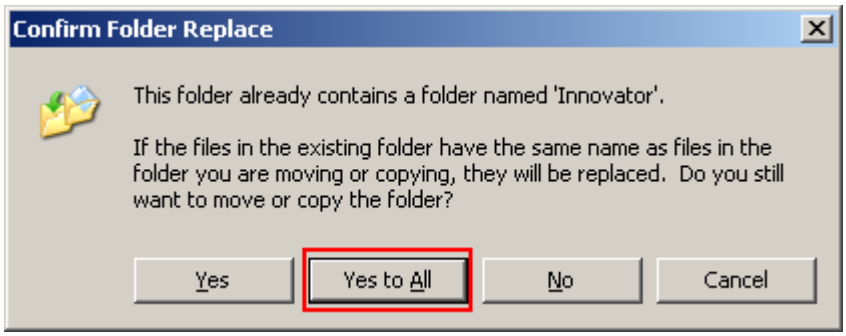


12. After the import is complete, **disable** the Super User login.

Manual Installation Option

1. Download the **Aras ProAppDesigner** 14 CD image from the Aras File Sharing site, in the Aras ProAppDesigner Releases folder, and unzip the file on the local computer.
2. Copy the \ `Files\Innovator\Client` folder to the root of your installation directory. When prompted, **overwrite** the existing `\Innovator\Client` folder and all of its contents.

It is **recommended** that this step be performed by an administrator on the server.



3. Copy the `\Files\ConversionServer` folder to the root of your install directory, overwriting the existing `\ConversionServer` folder and all its contents.

   It is recommended that this step be performed by an administrator on the server.

4. If any of the below DLLs are available in `\Innovator\Server\bin` or in ConversionServer\bin folder, delete them
   - Prorigo.Protrak.ExpressionEvaluatorStd.Impl.dll
   - Prorigo.Protrak.ExpressionEvaluatorStd.dll
   - Prorigo.Plm.PresentationGenerator.dll
   - Prorigo.Plm.PresentationGenerator.Aras.dll
   - Prorigo.Plm.Logging.Framework.dll
   - Prorigo.Plm.DocumentGenerator.dll
   - Prorigo.Plm.DocumentGenerator.Aras.dll
   - Prorigo.Plm.DiscussionPanel.dll
   - Prorigo.Plm.Core.dll
   - Prorigo.Plm.Aras.BackgroudCronJobFramework.dll
   - Prorigo.Plm.AppStudio.RuleEngine.Impl.dll
   - Prorigo.Plm.AppStudio.RuleEngine.dll
   - Prorigo.Plm.AppStudio.IntegrationFramework.dll
   - Prorigo.Plm.AppStudio.dll
5. From the root of your code tree, navigate to `\Innovator\Server` and edit the `method-config.xml file` .
   1. Add below entries as shown:

      ...

      ```
      <!-- ProAppDesigner DLLs : Start -->
      <name>$(binpath)/Prorigo.Plm.Aras.dll</name>
      <name>$(binpath)/Prorigo.Plm.Aras.AppStudio.dll</name>
      ```

```
<name>$(binpath)/Prorigo.Plm.Aras.TDP.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.SubscriptionFramework.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.DocumentGenerator.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.BackgroundCronJobFramework.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.RuleEngine.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.PresentationGenerator.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.IntegrationFramework.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.IntegrationFramework.Connector.dll</n
<name>$(binpath)/Prorigo.Plm.Aras.DiscussionPanel.dll</name>
<name>$(binpath)/Prorigo.Plm.Aras.Logger.dll</name>
<name>$(binpath)/Prorigo.Plm.LicensingFramework.dll</name>
<!-- ProAppDesigner DLLs : End -->
```

</ReferencedAssemblies>

2. Save the changes to the method-config.xml file and close it

Important

If any of the DLLs mentioned in the DLLs to be deleted section, please delete the corresponding entry from method-config.xml file.

6. From the root of your code tree, navigate to `\Innovator\Client` and edit the `InnovatorClient.config` file.
   1. Change the value of the filesRevision attribute. The value should be changed from std to 2, 3, 4, etc

```
<oauth configSource="OAuth.config" />
<cachingModule moduleEnabled="true" filesRevision="2" />
<staticContent>
```

...

   2. Save the changes to the `InnovatorClient.config` file and close it.
7. From the root of your code tree, open `ConversionServerConfig.xml` file.
   1. Add below entries in Converters.

```
<Converters>
<Converter name="bcj_CRONJOBS"
type="Prorigo.Plm.Aras.BackgroundCronJobFramework.BCJ_Converter,
Prorigo.Plm.Aras.BackgroundCronJobFramework" />
<Converter name="bcj_Background_Job_Conversion_Type"
type="Prorigo.Plm.Aras.BackgroundCronJobFramework.BCJ_Converter,
Prorigo.Plm.Aras.BackgroundCronJobFramework" />
```

```
</Converters>
```

...

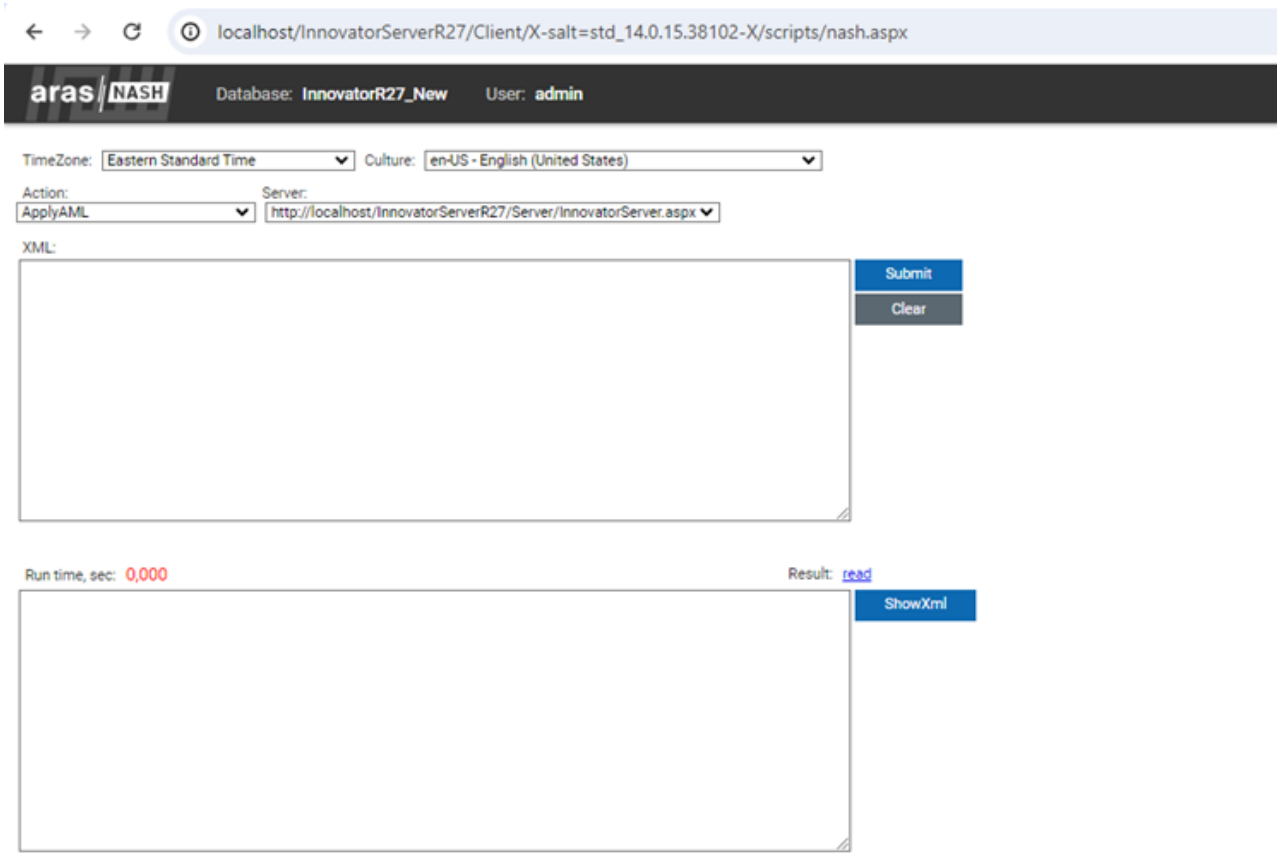
2. Save the changes to the `ConversionServerConfig.xml` file and close it.

8. Restart the World Wide Web Service (IIS).

9. Execute **Pre Import Scripts**

   Before you install Aras ProAppDesigner, execute the following scripts:

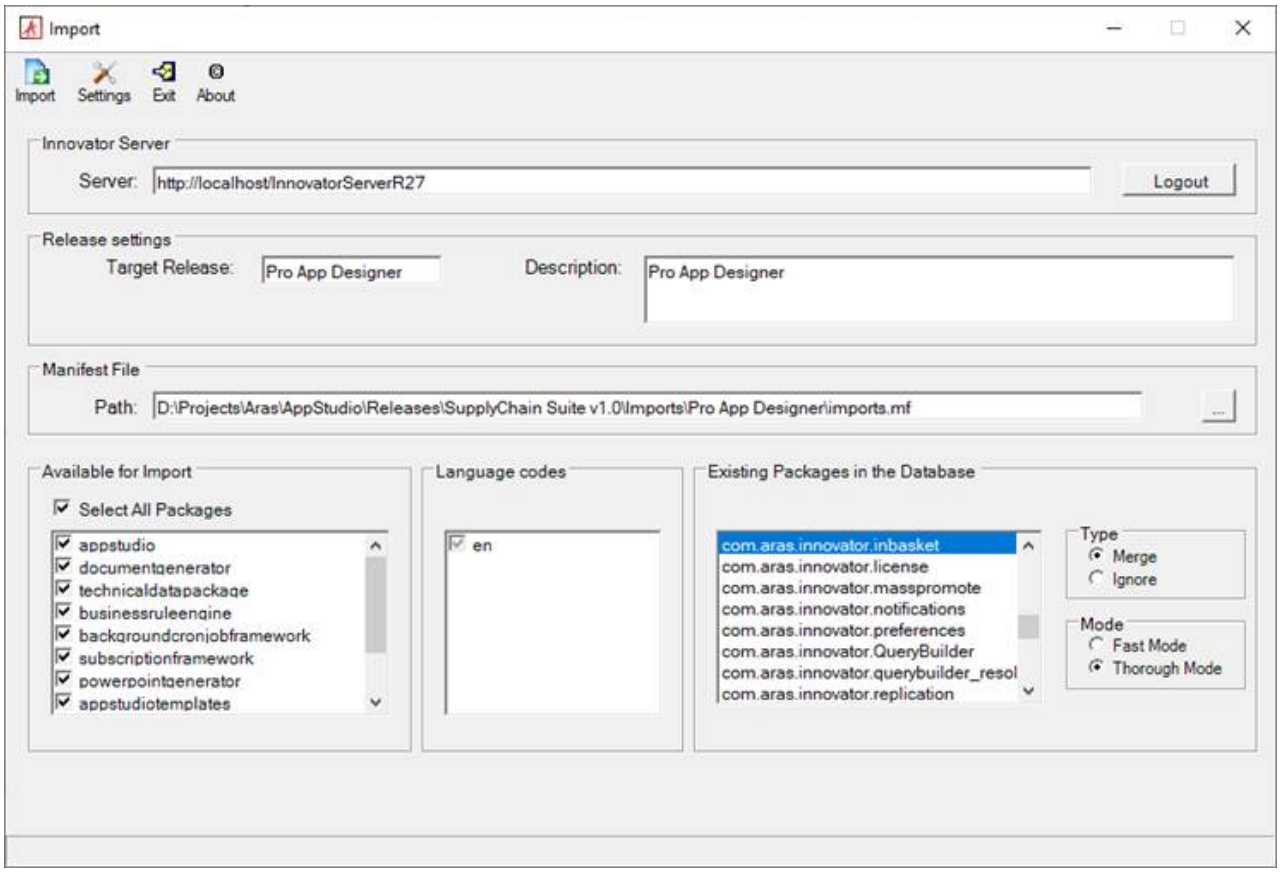
   1. Navigate Aras Nash tool using URL http://localhost/InnovatorServer/Client/**scripts/nash.aspx**
   2. Login in Nash using **admin** user.



   3. In the package go to folder PreImportAMLScripts\Manual Scripts, open each file in below sequence, copy the content in XML form and click on Submit.
      - 04500_TemplateDeploy_PreAML.xml

10. Complete the Installation of ProAppDesigner: Import the **ProAppDesigner** database package with the Package Import Export Utilities (For more information using this tool, please see the *Aras Innovator - Package Import Export Utilities* documentation.)

1. You will need to **enable** the Super User login to perform this step.
2. Browse to the \PackageImportExportUtilities\Import\ folder and run the Import.exe file.



3. Enter the following information:

   - **Server** = The connection URL for Aras Innovator
   - **Database** = The target Aras Innovator database (login screen)
   - **Username** = root (login screen)
   - **Password** = Password for “root” login (Default is “innovator”) (login screen)
   - **Target Release** = ProAppDesigner
   - **Description** = ProAppDesigner
   - **Manifest File Path** = The manifest is found in the unzipped patch folder in \Aras ProAppDesigner CD Image\Imports\ProAppDesigner\imports.mf
   - **Available for Import** = Check all packages
   - **Type** = Merge
   - **Mode** = Thorough Mode

4. Click the Import button.

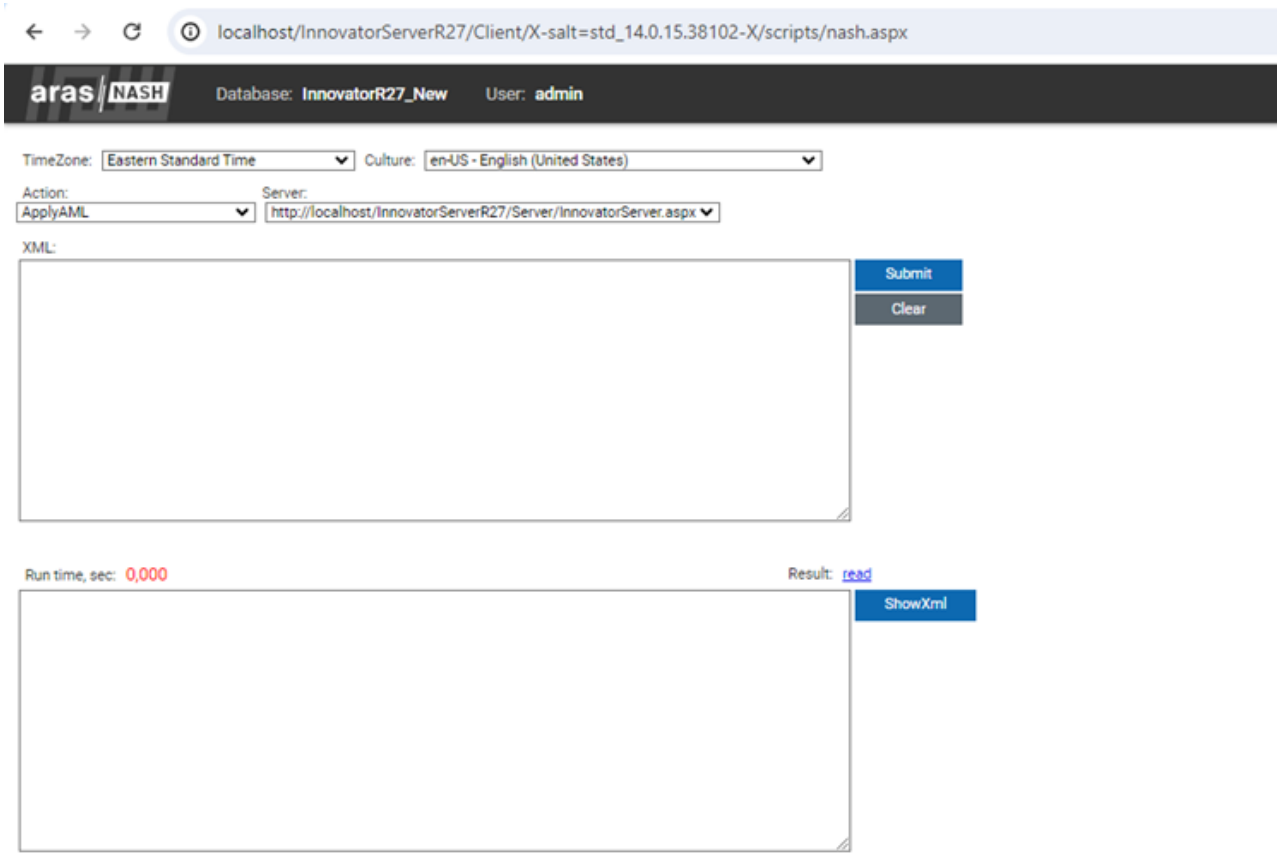
11. **Disable** the Super User login.

12. Execute **Post Import Scripts**

    After you install Aras ProAppDesigner, execute the following scripts:

    1. Navigate Aras Nash tool using URL http://localhost/InnovatorServer/Client/**scripts/nash.aspx**

    2. Login in Nash using **admin** user.



    3. In the package go to folder PostImportAMLScripts\Manual Scripts, open each file in below sequence, copy the content in XML form and click on Submit.
        - 05000_TemplateDeploy_PostAML.xml

Confirming the Installation

Use the following procedure to check if your database is updated correctly:

1. og in to Aras Innovator as an administrator.
2. From the TOC, select **Administration** → **Variables**.
3. Confirm the following sets of variables are listed:

Pro App Designer
14.14.22

If at any time the installation fails, revert to your backups and contact Aras Support at support@aras.com.

Activating ProAppDesigner

You require any one of the listed feature licenses for ProAppDesigner. Once you have received the keys, activate them from **User Menu** → **Activate Feature**.

1. Aras.ProAppDesigner
2. Aras.SupplierManagementAndPortal

Setting Required Variables

Make sure to create and set values for below variables:

Variable Name	Value Example	Remarks
Prorigo.Plm.InstanceURL	http://localhost/InnovatorServer	This variable is used by subscription framework to send the item link in mail body. If not created or left empty, the mail body will not have correct item link.

Aras Technical Data Package - User Guide

Introduction

Purpose

This guide describes how to use Aras Technical Data Package for external users.

Scope

This document provides guidance on the configuration and functionality of the Aras Technical Data Package application.

Target Audience

This document is intended for external users with limited access to data.

Guide Organization

This document describes how to use the Aras Technical Data Package application. This User Guide is organized into the following sections:

SECTION 1: Introduction	This section outlines the purpose and scope of this User Guide, as well as its intended target audience. It supplies a brief overview of each section to help users in navigating the guide.
SECTION 2: Introduction to Technical Data Package (TDP)	This section outlines the purpose of the Technical Data Package (TDP) and provides an overview of the key functionalities available within the Aras Technical Data Package application.
SECTION 3: TDP Template	This section outlines how to select and generate TDP Excel Template which are the fundamental building blocks in TDP.
SECTION 4: Export TDP	This section outlines how to create custom action and export TDP.
SECTION 5: Import TDP	This section outlines how to create custom action and import TDP.

Introduction to Technical Data Package (TDP)

Purpose of Technical Data Package

The Aras Technical Data Package (TDP) application enables offline access for external suppliers by allowing specific data and files to be securely shared. Through the Export TDP action, users can export items, relationships, and associated files into a ZIP archive. This package includes an Excel file containing structured data across multiple worksheets, with columns representing properties from ItemTypes, RelationshipTypes, and Files. The package is generated on the server based on user actions, lifecycle promotions, or scheduled jobs. When updates are made, the Import TDP action allows users to re-import the modified data and files back into Aras Innovator. The Excel template used for package generation is selected according to predefined configuration rules.

TDP Template

Introduction

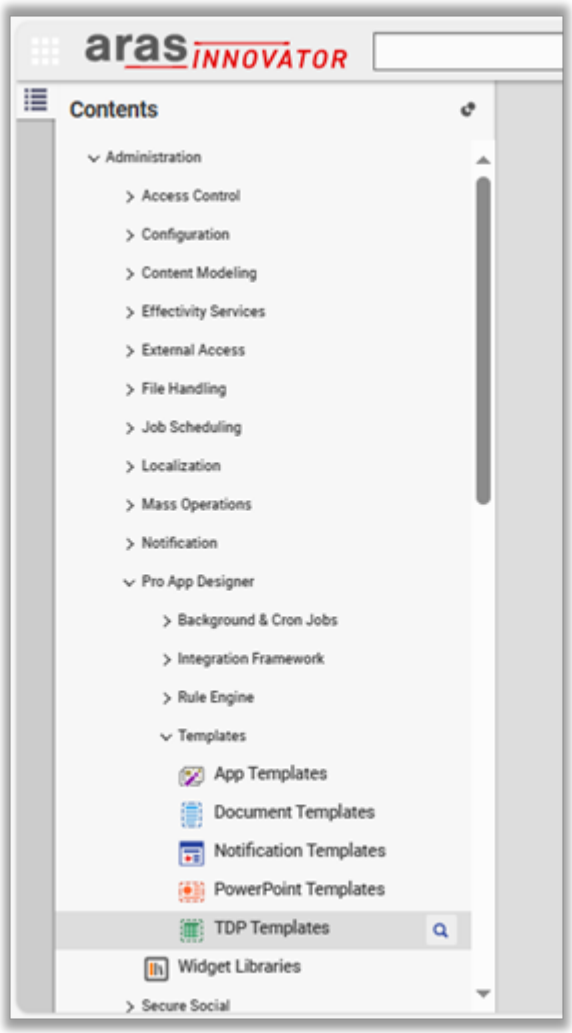
The Technical Data Package (TDP) is generated based on a configured Excel template that includes multiple worksheets. Each worksheet contains columns mapped to properties from ItemTypes, RelationshipTypes, and Files, ensuring structured and comprehensive data representation. Only one TDP template item can be associated with a specific ItemType, maintaining a clear and consistent configuration. However, each TDP template can include multiple Excel files, allowing flexibility in how data is organized and presented within the package.

TDP Template Management

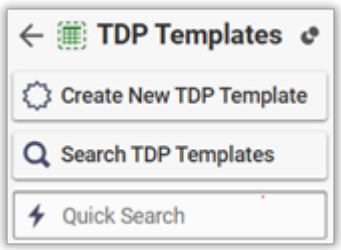
Create Template

The following steps outline the process of creating **TDP Template**:

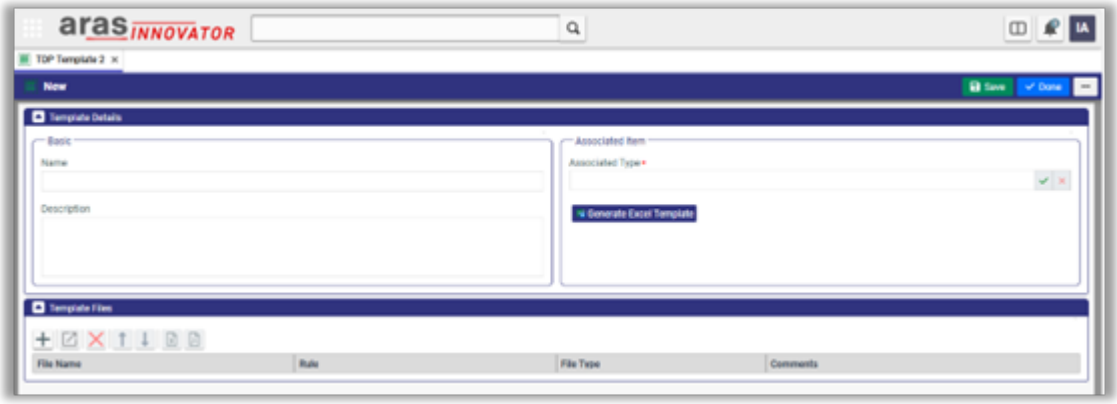
1. From the Table of Contents select **TDP Templates**.
   (Go to TOC → Administration → Pro App Designer → Templates →TDP Templates.)



2. The **TDP Templates Table of Contents** appears. Click **Create New TDP Template**.



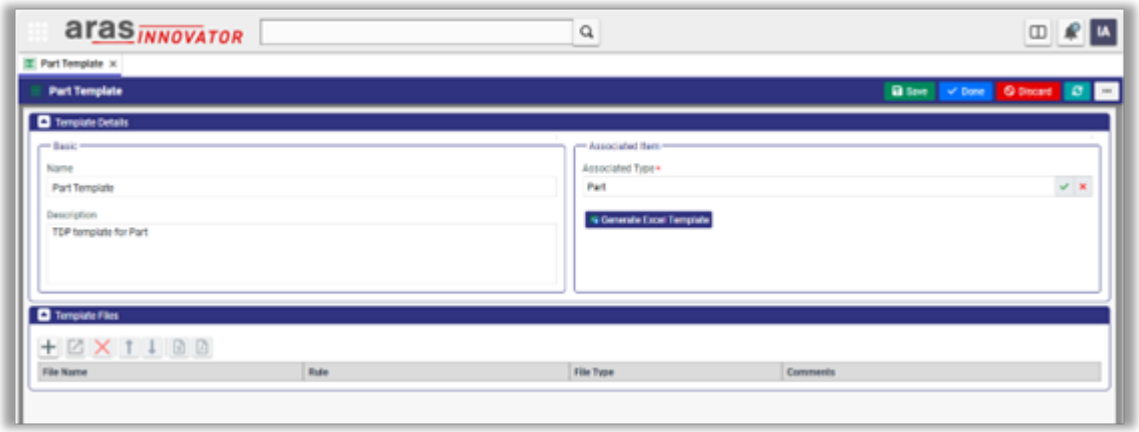
3. Fill in the required properties, add the appropriate template file, and then click Save or Done to complete the process.



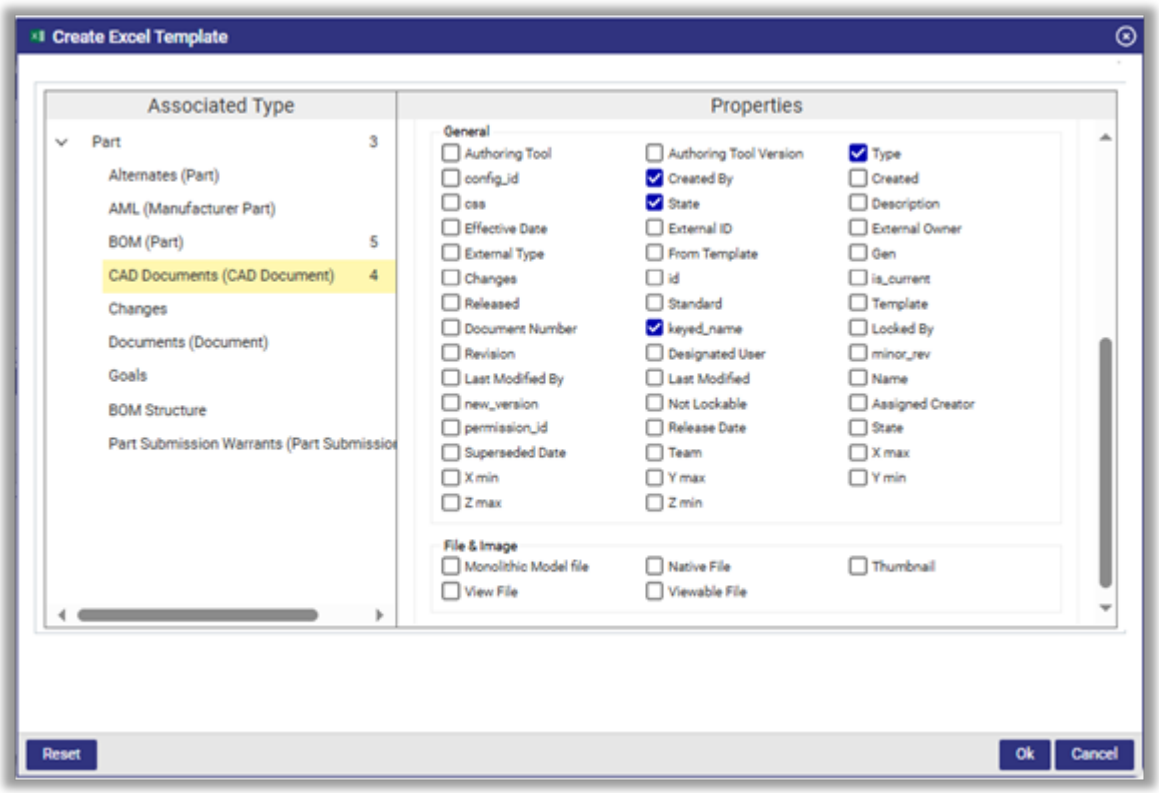
Auto-Generate TDP Template

The steps below detail the process for automatically generating a TDP template file:

1. Click on **Generate Excel Template** button under Associate Item group.

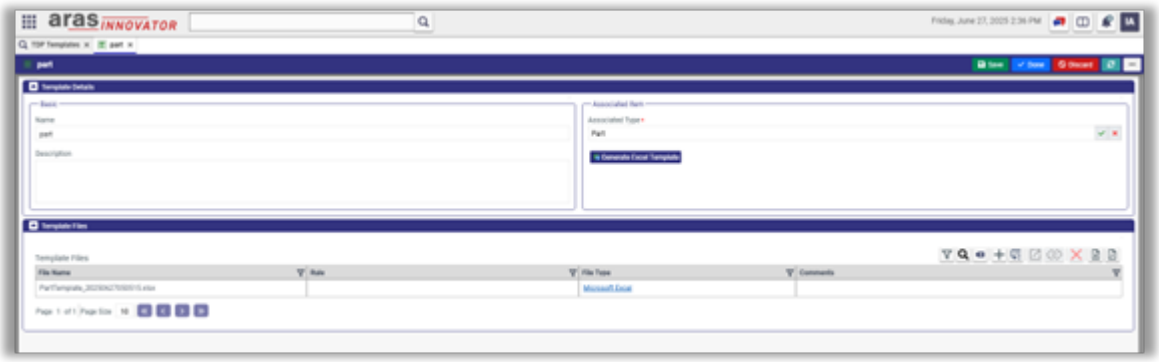


2. The **Create Excel Template** dialog will open.



3. Select the properties and click on **Ok** button.

4. New **Template Files** relationship will create.



Excel Template Selection

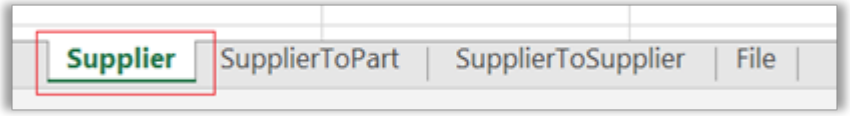
Each TDP template can have multiple Excel files attached. During the Export TDP operation, the appropriate Excel file is automatically selected based on predefined configuration rules:

1. If an Excel file within a TDP template does not have any rule configured, it will be selected by default during the Export TDP operation. In cases where multiple Excel files lack rules, the system will automatically select the first available file.
2. If all Excel files within a TDP template have rules configured, the system will select the first file that satisfies its rule during the Export TDP operation. If none of the files meet their respective rule conditions, the Export TDP operation will be aborted.

Template File Assumptions

To ensure successful and accurate export of a Technical Data Package (TDP), the Excel files attached to the TDP template must adhere to specific formatting and structural guidelines.

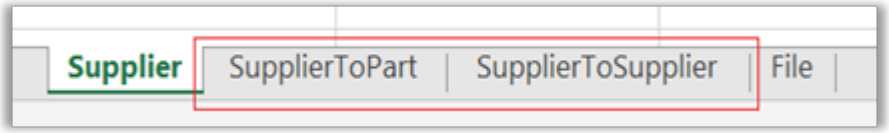
1. The name of the first worksheet must exactly match the name of the context ItemType. The property columns names should be exactly same as ItemType property names.



2. The final worksheet in the Excel file must include fully qualified paths to the file locations required for the Technical Data Package (TDP). These file paths can originate from ItemType properties, Relationship properties, or File relationships. If the file location is defined through a File relationship, the "Property" column in the worksheet may be left blank. An example of how to describe file location is shown below.

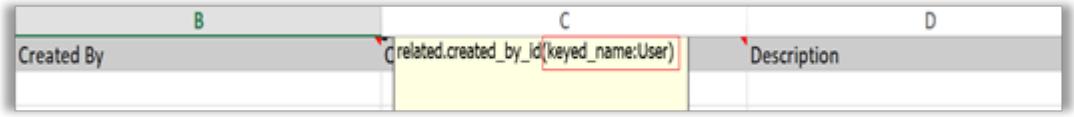
FileItemTypePath	Property
Supplier/SupplierToPart/Part/Part Document/Document/Document File/File	
Supplier/SupplierToPart/Part/Part CAD/CAD	native_file
Supplier/SupplierToPart/Part/Part CAD/CAD	viewable_file
Supplier/SupplierToPart/Part/Part CAD/CAD/CADFiles	attached_file

3. Intermediate worksheets can be added as needed to export data related to relationships and associated items. Each of these worksheets must be named exactly after the corresponding Relationship name to ensure proper mapping. Additionally, the column headers within each intermediate sheet must match the property names defined in the Relationship.

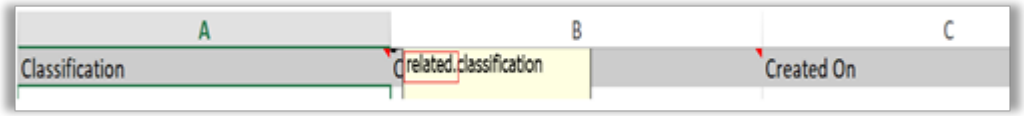


4. In each worksheet, the columns should display the property labels, while the corresponding property names must be included as comments within the header cells.

5. If the property type is item then add itemtype name in comment.

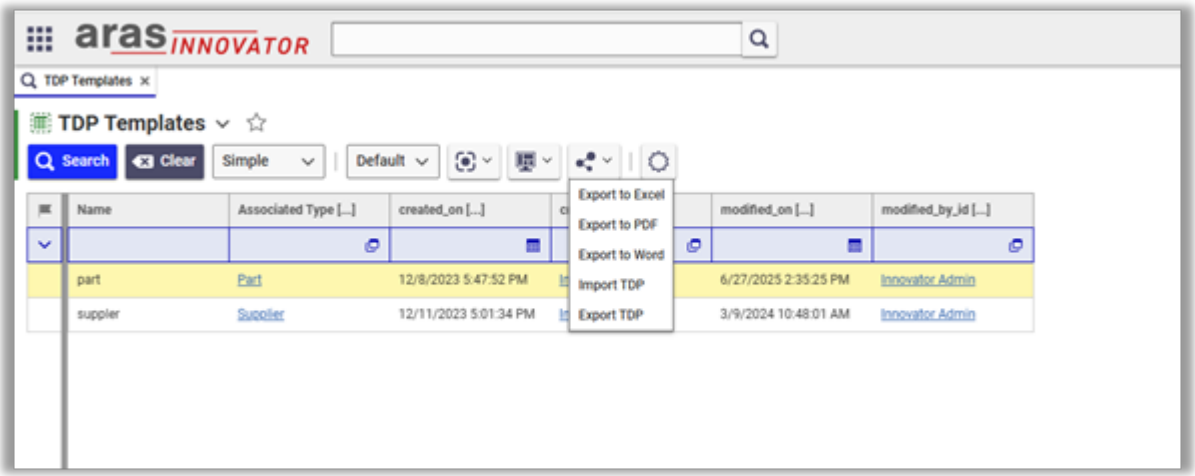


6. The RelatedItem properties can be exported by adding the prefix “related.” to the property name.

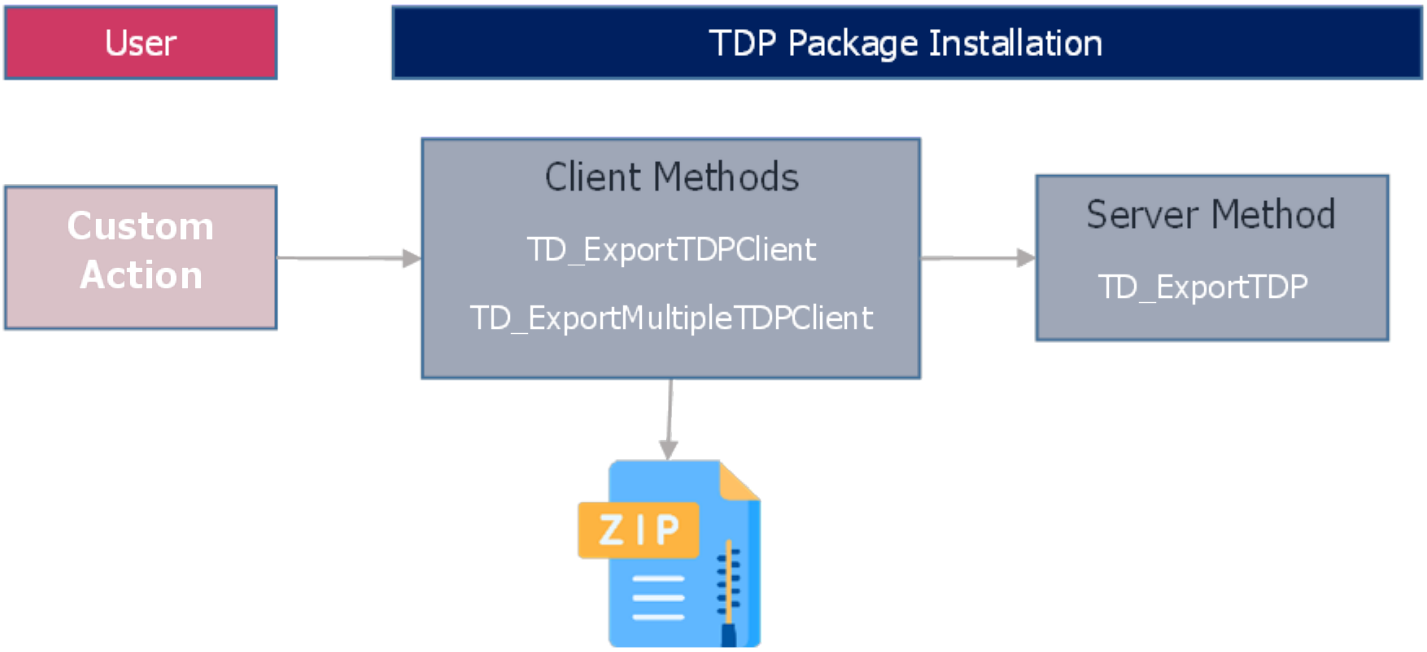


Export TDP

The Export TDP action can be configured for any ItemType and made available on both the Item Details page and the Item Search page.



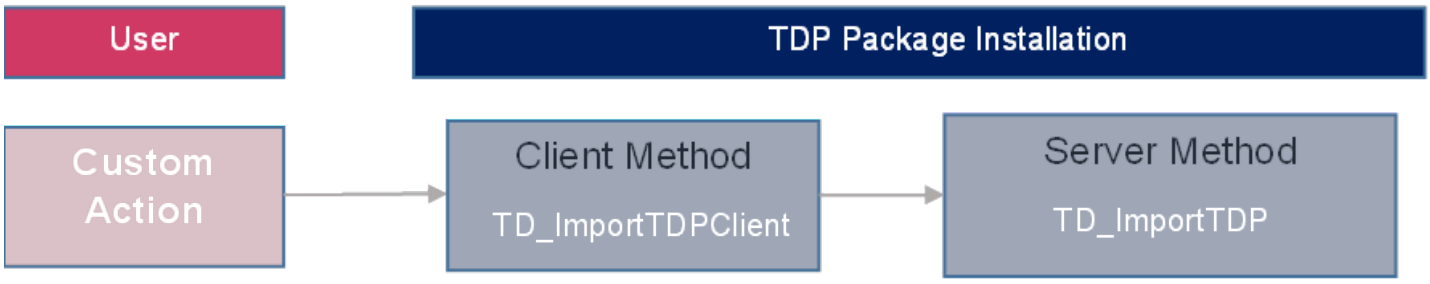
Once the standard Export TDP UI action is configured by the user, the associated client method can invoke server-side methods provided with the TDP installation to generate the Technical Data Package. The server method processes the request and returns the TDP item to the client, which then uses standard Aras functionality to download the package to the user's default local download location. The exported package is delivered as a ZIP file containing an Excel document with all relevant items and relationships data, along with any associated files. When exporting from the Search Grid, users can select multiple items, and the system will compile all related data and files into a single Excel file and ZIP archive for export.



The Excel file generated as part of the Technical Data Package (TDP) can be edited offline, allowing users to update data as needed before re-importing it into Aras Innovator. To prevent accidental modifications, the File worksheet within the Excel file is password-protected. The default password can be changed by updating the value stored in the td_TDPPassword variable. Additionally, each worksheet contains certain columns highlighted in grey—these are essential for the re-import process and must not be edited or deleted to ensure successful data integration.

Import TDP

Using the 'Import TDP' action, the updated package can be re-imported into Innovator. Users must configure the 'Import TDP' action within the Search page of any ItemType.



Aras Word PDF Generator - User Guide

Introduction

Purpose

This User Guide describes how to use the Aras Word/Pdf Generator for external users.

Scope

This document provides instructions for configurations and functionality of Aras Word/Pdf Generator application.

Target Audience

This document is intended for external users with limited access to data.

Guide Organization

This document describes how to use the Aras Word/Pdf Generator application. This User Guide is organized into the following sections:

SECTION 1: **Introduction**	This section defines the purpose and scope of the User Guide, identifies its intended audience, and provides a concise summary of each part to assist users in navigating the document effectively.
SECTION 2: **Introduction to Aras Word/pdf Generator**	This section explains the purpose of the Aras Word/PDF Generator and provides an overview of its key features. It also includes guidance on creating templates.

Introduction to Word/PDF Generator

The Aras Word/PDF Generator automates the creation of Microsoft Word and PDF documents based on user selections made through wizard pages. Documents are generated using pre-configured Word templates and do not require any plugins or Office connectors. Generation occurs server-side and can be triggered by user actions, lifecycle promotions, or scheduled jobs.

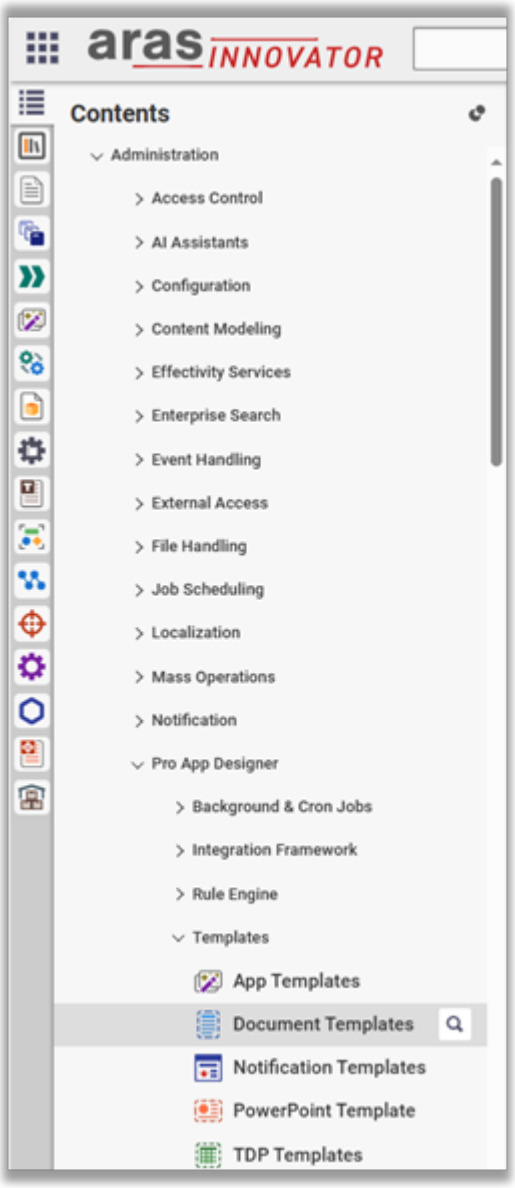
Templates are designed with content controls mapped to properties of ItemTypes and RelationshipTypes, allowing dynamic data population. Documents can also be assembled from existing files stored within Aras Innovator. Templates may include repeating content controls, which are linked to data sources derived from relationship collections or server method outputs.

Additional features include watermarking based on lifecycle states and rule-based template selection for document generation.

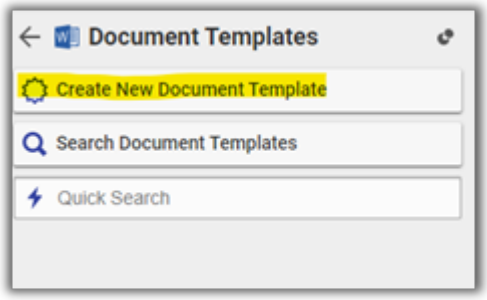
Document generation can be initiated by any user action or system event. The process utilizes the template associated with the relevant ItemType.

Document Generator Template Configurations

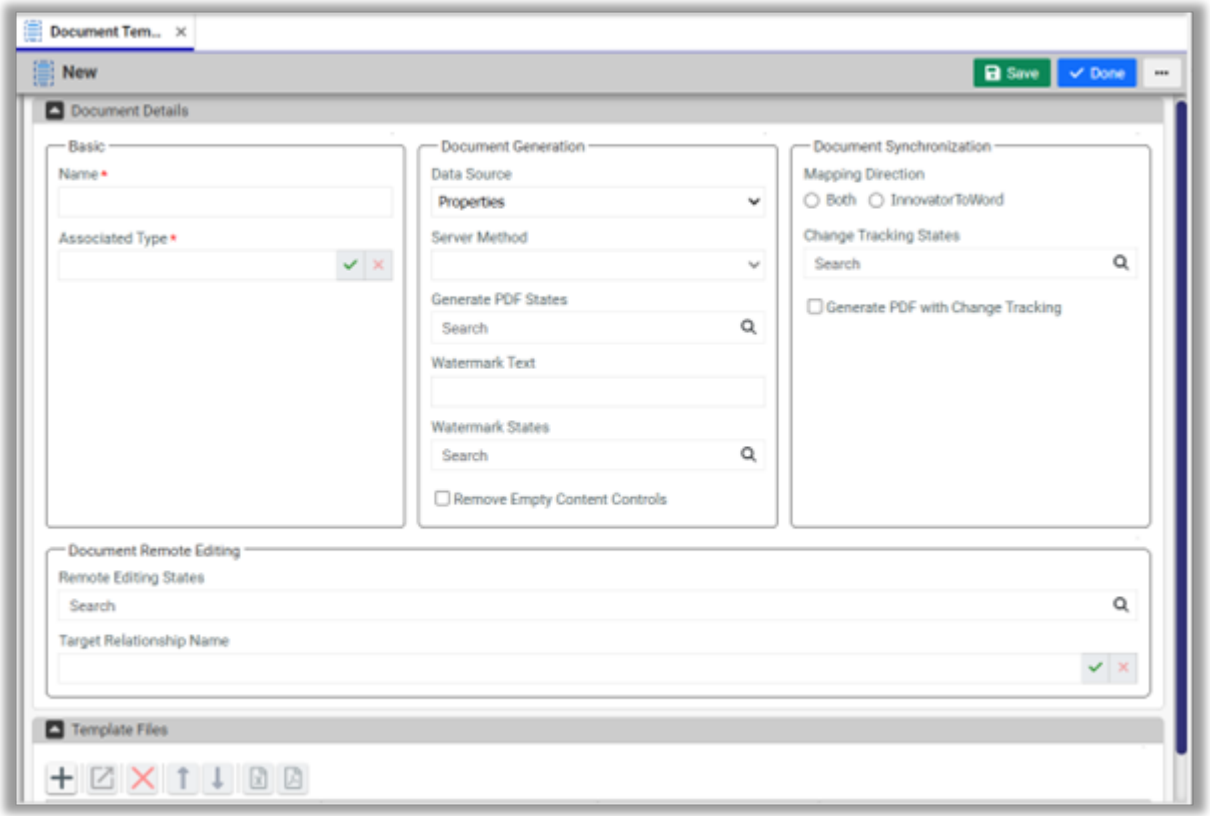
1. Go to TOC → Administration → Pro App Designer →Templates → Document Templates.



2. Click on Create New Document template.



3. Enter the necessary information, save your work, and add the template file accordingly.



Following are the details for Document Template properties:

Basic:

- **Name**: Template Name is a required property used to identify the document template. A valid name must be provided in order to save the template.
- **Associated Item**: Context Item specifies the ItemType for which the document template is being created. This is a required property and must be defined in order to save the template.

Document Generation:

- **Data Source**: Supported values are 'Properties' (selected by default, Content variables are mapped using the properties of the selected ItemType), 'Server Method' (Content variables are mapped using the output of a specified server method).
- **Server Method**: Server Method Selection option becomes available when 'Server Method' is chosen as the data source for content mapping. All server methods attached as Server Events to the associated ItemType will be listed for selection. The selected method will be used to map content variables within the document template.
- **Generate PDF States**: States list for which PDF will be generated.
- **Watermark States**: Specifies the list of states for which watermarks should be applied when generating PDF documents.
- **Watermark Text**: Text to be used for watermarking. This attribute is applicable only if the watermark States attribute is defined.
- **Remove Empty Content Control**: Determines whether empty content controls should be removed from the generated PDF document.

Document Synchronization:

- **Mapping Direction**: Supported values are 'Both' (Content variables will be mapped from Innovator to Word and Vice versa), 'InnovatorToWord' (Content variables mapping will be only from Innovator to Word)
- **Change Tracking States**: States list for which mapping will be done after changes.
- **Generate PDF with Change Tracking**: Determines whether a change-tracked PDF should be generated, based on whether any modifications have occurred.

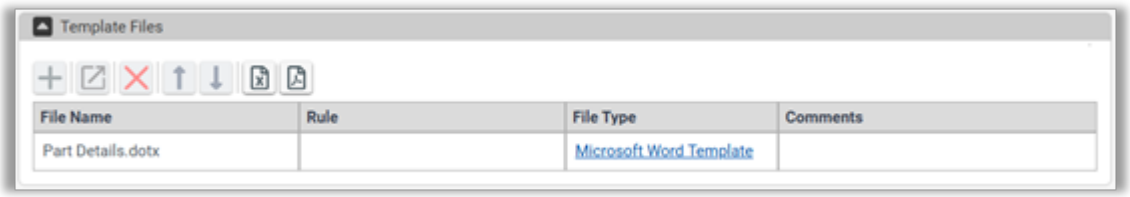
Document Remote Editing:

- **Remote Editing States**: States list for which remote editing is allowed.
- **Target Relationship Name**: Name of relationship where the files will attach after processing.

Template Selection

Each Document Generator template can have multiple Microsoft Word files associated with it. During the document creation process, the appropriate Word file is automatically selected based on predefined rules:

- If a Word file is attached to the template without any selection rule configured, it will be chosen by default during document generation. In cases where multiple Word files lack rules, the system will automatically select the first available file.
- If all attached Word files have selection rules defined, the system evaluates each rule in order. The first Word file that passes its validation criteria is selected for document generation. If none of the files satisfy their respective rules, the document generation process is aborted.



Word Template Creation

Enabling Developer Tab in Microsoft Word

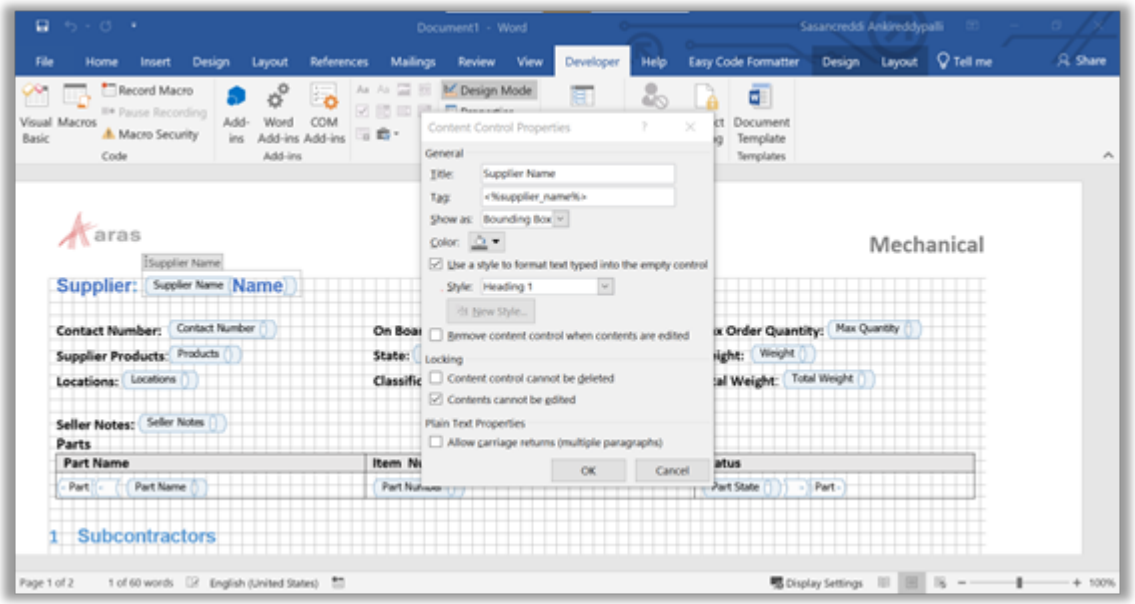
In Microsoft Word, the Developer tab provides access to content controls, which are essential for creating dynamic templates. However, this tab is not visible by default. To enable it, navigate to **File > Options > Customize Ribbon**. In the right-hand panel of the customization window, locate and check the Developer checkbox, then click OK. Once activated, the Developer tab will appear in the ribbon, offering a variety of content controls such as text boxes, drop-down lists, and combo boxes. Each of these controls comes with configurable properties that allow users to tailor their behavior and data bindings to suit specific template requirements.

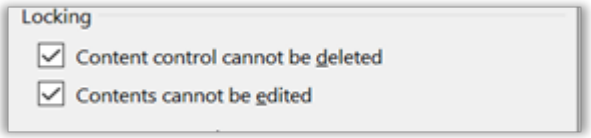
The WordSynchronizer application utilizes three types of content controls in Microsoft Word: Plain Text, Rich Text, and Date Picker. Each control type serves a specific purpose in template design and data mapping. The table below outlines the content control types used in WordSynchronizer along with their general descriptions:

Type	General Description
Plain Text	A plain text content control is designed to hold only plain text, either in a single paragraph or across multiple paragraphs. It cannot contain complex elements such as tables, images, or other embedded content controls. It is used for Aras Innovator → MS Word mapping and MS Word → Aras mapping as well.
Rich Text	A rich text control allows users to enter and format text with styles such as bold, italic, and underline. It can also contain complex elements like tables, images, and even other content controls. However, in the context of MS Word → Aras, this control is only supported for content flowing from Word to Aras. To store the HTML-formatted content in Aras, an additional text property must be configured to handle the rich text data. Please note that if you have summary as a Property and you want to have its html equivalent property it has to be 'summary_html'.
Date Picker	Displays a calendar from which the user can click a date. It is used for Aras Innovator → MS Word mapping and MS Word → Aras as well.

Repeating	This control allows the duplication of its entire contents, including nested content controls. It is typically used to repeat structured elements such as paragraphs or table rows. null This content control is available onwards MS Office 2013.

Using Controls: Plain Text Content Control with Locking Behavior





Content control cannot be deleted

1. This restricts the user to delete the content control.
2. Recommended to set this option ‘ON’ once template design is freeze.

Contents cannot be edited set to True [Used for Aras à MS Word mapping]

When the “Content Cannot be Edited” option is True, it prevents users from modifying the content externally. This setting is used to insert property values from Aras Innovator directly into the Word document at designated placeholders. For instance, if the tag <%supplier_name%> is used and the option is set to True, the system will automatically substitute this tag with the value of the supplier_name property from Aras Innovator.

It’s important to note that this configuration supports a one-way mapping from Aras Innovator to Microsoft Word. Whenever the goal is to retrieve and display a property value from Aras Innovator in

a Word document, this option should always be set to True. For this behavior, only Plain Text and Date Picker content controls are supported—Rich Text controls are not compatible.

Contents cannot be edited set to False [Used for MS Word à Aras mapping]

When the “Content Cannot Be Edited” option is False, users can enter or modify content within the control, and the updated value will be saved to the corresponding property of the Aras Innovator item during the Save operation. This type of mapping supports Rich Text, Plain Text, and Date Picker content controls. It’s important to note that this mapping is only valid for direct properties of the associated item and does not apply to relationship properties or derived data.

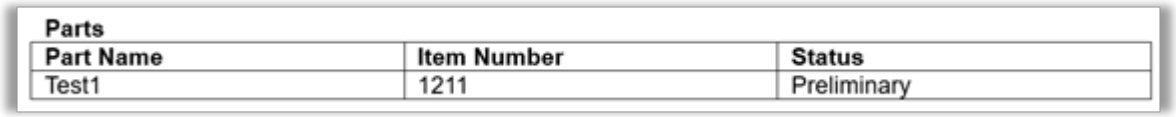
Using Controls: Repeating Text Content

Repeating Content Controls allow entire tables or specific sections within a Word document to be repeated multiple times, each populated with a different dataset. This functionality relies on input provided in XML format, leveraging Microsoft Word's custom XML feature to dynamically generate content.

For more details on Word's custom XML capabilities, please refer to the following link:

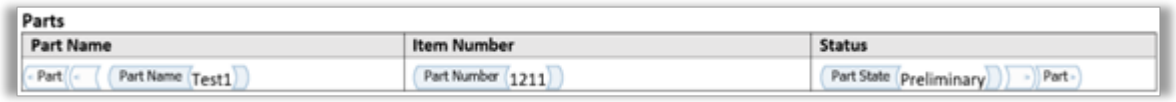
https://docs.microsoft.com/en-us/visualstudio/vsto/how-to-add-custom-xml-parts-to-document-level-customizations?view=vs-2019

The WebSynchronizer application processes JSON input received from an Aras Innovator Server Method, which is configured as the data source property of the Document Template item. When the associated content control in the Word template is a repeating control, WebSynchronizer automatically converts the JSON data into XML format. This XML is then used to populate the repeating sections in the document. For example, in the case of a table, the data row will be repeated as many times as there are entries in the XML dataset.



Parts

Part Name	Item Number	Status
Test1	1211	Preliminary



Parts

Part Name	Item Number	Status
Part - Part Name Test1	Part Number 1211	Part State Preliminary - Part

Please refer following XML for condition table which is returning single condition. Rows will be repeated number of times for `<SupplierToPart.related_id>` tag in XML.

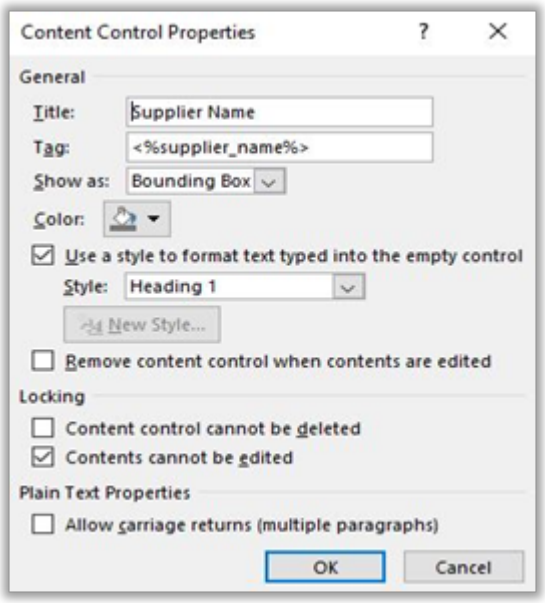
```
<root>
<SupplierToPart.related_id>
<name>Test1</name>
<item number>1211</item number>
<state>Preliminary<state>
```

```
</SupplierToPart.related_id>
</root>
```

For Reference, please also find the relevant JSON below.

"\" SupplierToPart.related_id \"
:[{\"name\":\"Test1\"
,\"item_number\":\"1211\"
,\"state\":\"Preliminary \"}]"

Following image shows how repeating content control is mapped to <%SupplierToPart.related_id%> which is tag from XML.



Tag Naming Conventions and its meaning

In the above image tag value is given as <%supplier_name%>. Let us see what it means

1. <%..%> Indicates, processing at server end. Please note that it is mandatory to follow this convention.
2. <%**supplier_name**%> Here ‘supplier_name’ is actually the name of the Property of ‘Supplier’ Item Type. By this tag, application recognizes this property value to be fetched from Aras Innovator and substitute in the MS Word at relevant placeholder
3. <%**SupplierToPart.related_id.name**%> Here involves relationship.
    1. ‘SupplierToPart’: is the RelationshipType of Supplier Item Type
    2. ‘related_id’ is the property pointing to Part Item Type
    3. ‘related_id’ is the property pointing to Part Item Type

So, this tag value actually fetches the value of Property ‘name’ attached to ‘Part’ item type.

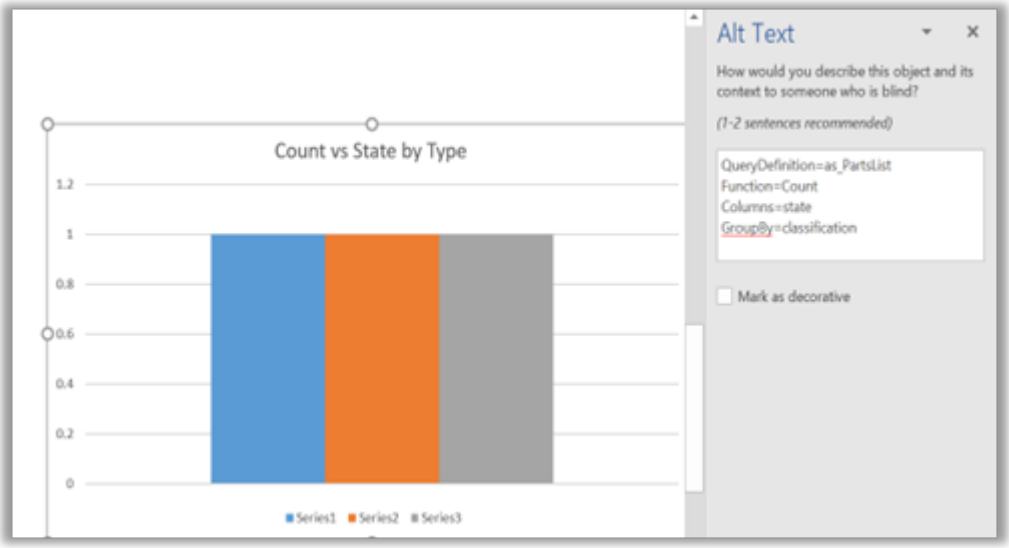
Using this naming convention, the server follows a specific sequence when processing a tag at server

1. Go to ‘SupplierToPart’
2. Search for related_id and get the Data Source value which is ‘Part’ in this case
3. Go to Part and access the ‘name’ property value.

Chart Support

Query Definition is used to retrieve data for chart generation in Aras Innovator. Refer following example snippet for fetching part status chart

QueryDefinition=as_PartsList
Function=Count
Columns=state
GroupBy=classification
This configuration will be provided as a “Alt Text” property of chart object in Word Template.



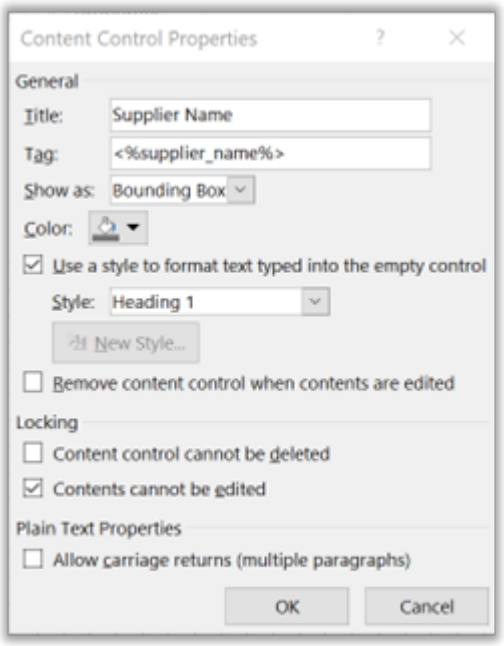
Mapping Content Control to an Item Type in Server Method

Let’s consider the ItemType "Part". If you want to fetch specific properties from this item, the following snippet demonstrates how to retrieve the name which is property of Part item. These retrieved properties are intended to be mapped to content variables in the Word document, as described in the previous section.

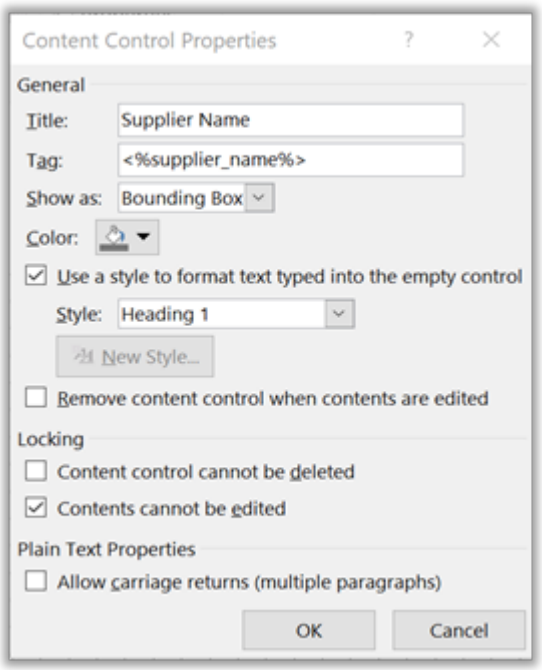
```
var partItemQuery = innovator.newItem("Part", "get");
partItemQuery.setID(partId);
partItemQuery.setAttribute("select", "name");
var partItem = partItemQuery.apply();
partObject.Add("name", partItem.getProperty("name "));
In case of multiple properties, please refer the code snippet below
var partItemQuery = innovator.newItem ("Part", "get");
partItemQuery.setID(partId);
partItemQuery.setAttribute("select", "name,item_number");
var partItem = partItemQuery.apply();
partObject.Add("name", partItem.getProperty("name"));
partObject.Add("item_number ", partItem.getProperty("item_number”));
```

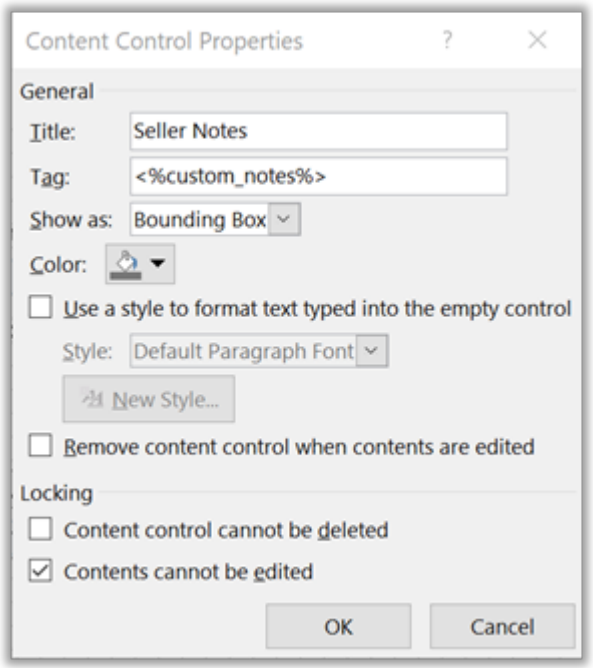
Mapping to remove empty content controls along with Label

1. Add label as plain text content control. In the properties dialog, check the option “Contents cannot be edited” .

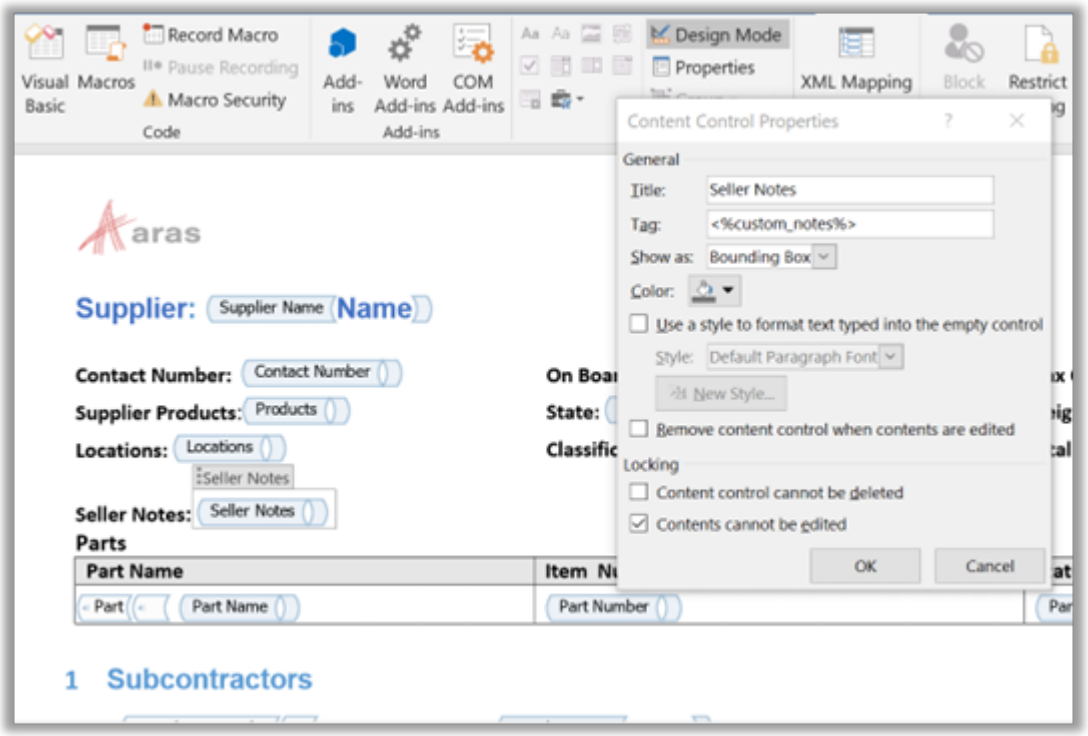


2. Label content control and data content control title should be same as shown below





3. Template designers can use line and spacing settings in Microsoft Word to add extra space after a content control. As shown in below image for Seller Note data content control.



Steps to edit placeholder text directly from 'Design Mode' and to preserve the original style

Click "in" the existing placeholder text as shown and modify the text to preserve formatting. Alternatively, you can also select the whole text and write yours

Clicking inside the placeholder text preserves its formatting. However, if you delete the placeholder text or begin typing at its start, the placeholder style will be lost..

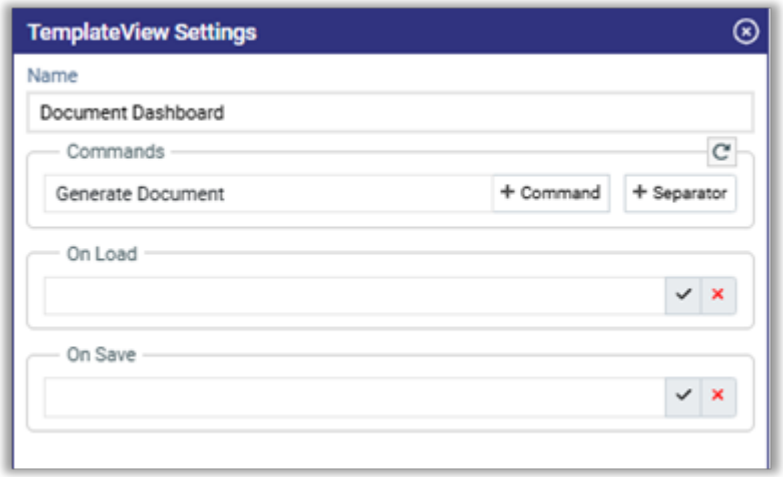
Notice that Starting at the beginning text is **'Black - Bold'**. Here the original placeholder text style is lost. To avoid this you can also select the whole text and write yours

Starting "in" the existing text preserve the placeholder text style. We can later delete the unwanted text

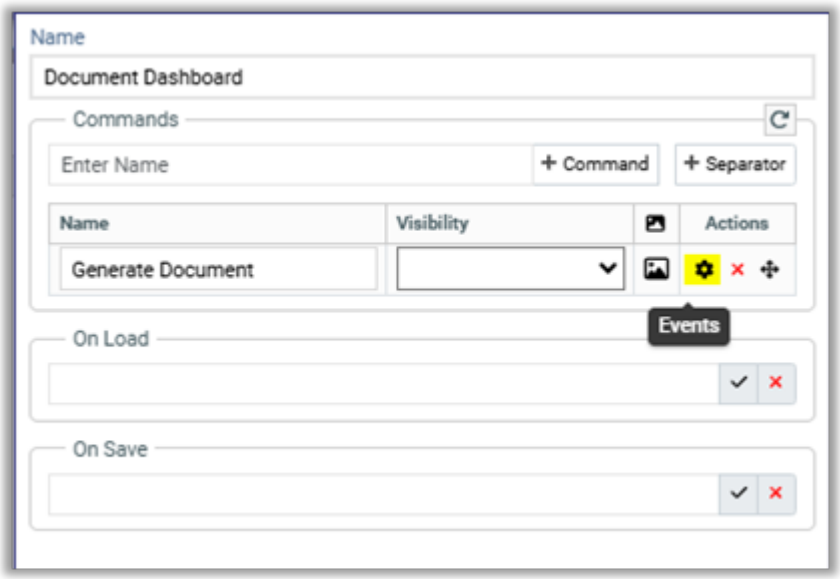
Configure Word Generator Command on Template View

To generate documents using this generator, it can be configured as a command on Item template with following steps

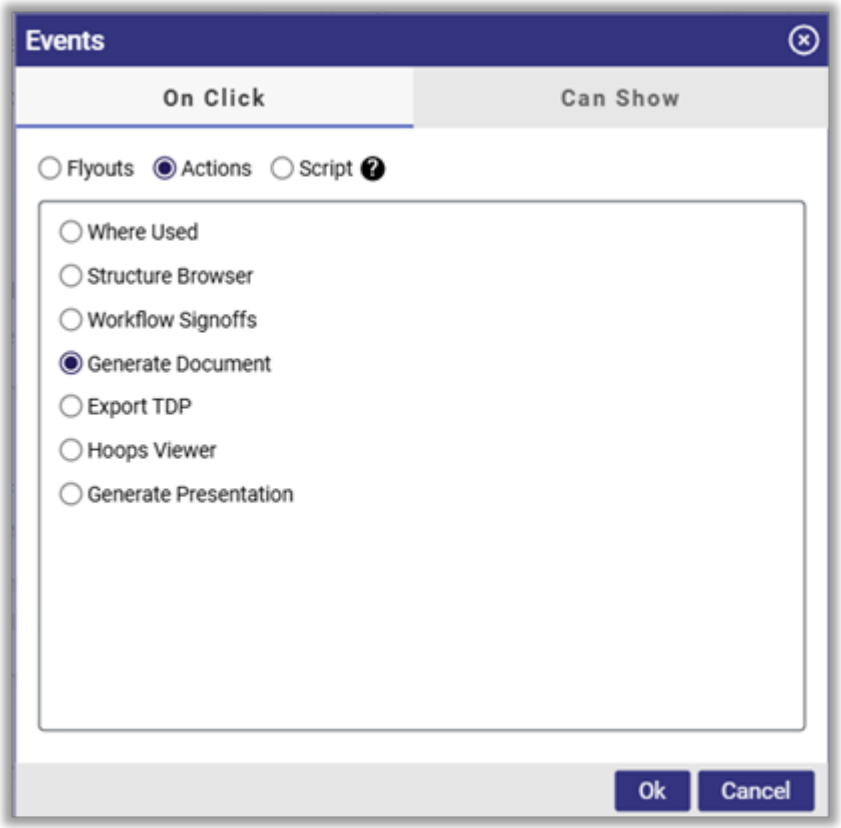
1. Go to TOC →Administration →Pro App Designer →Templates → App Templates → Open App Template and then Open Template View
2. Go to More →Settings → Add Command name (like below image) → Click on Command button.



3. Click on Events (highlighted in below image)

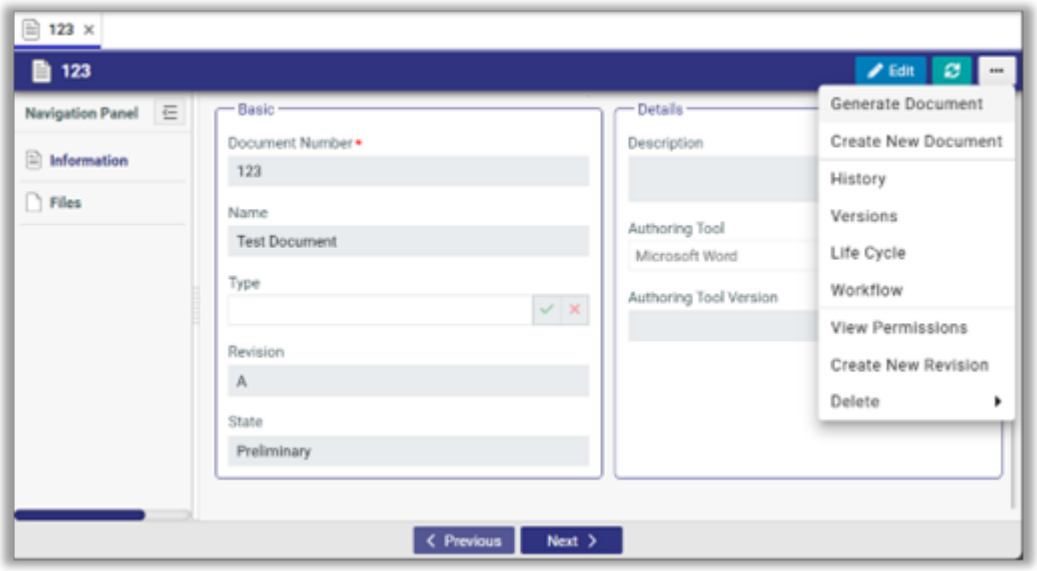


4. Select Actions and then Select Generate Document, click Ok.



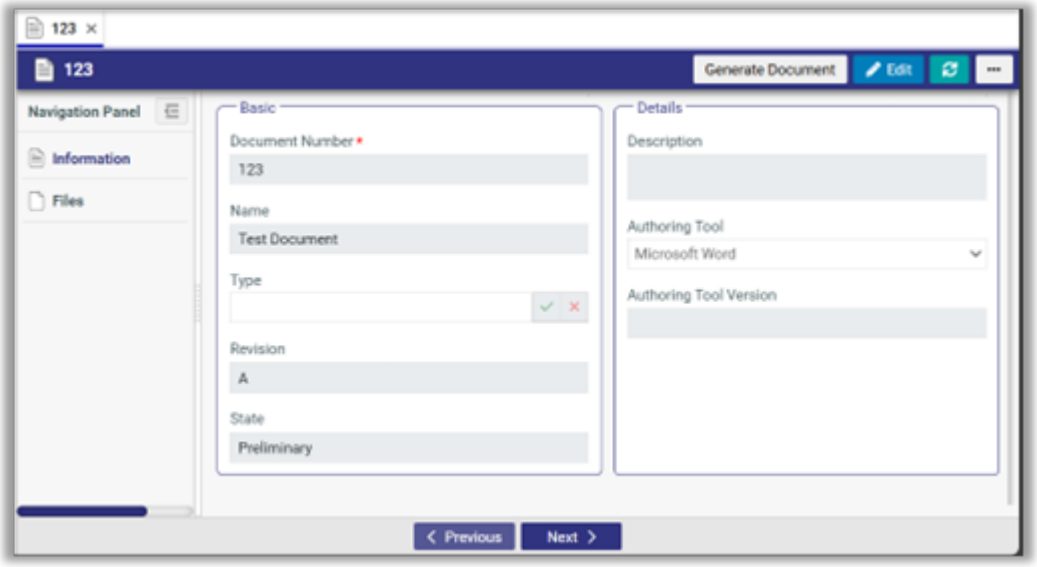
5. Add Visibility, then click on Ok.

6. Example of Commands with different visibility
   Visibility: Show In Dropdown
   eg:



Visibility: Show In ButtonGroup

eg:

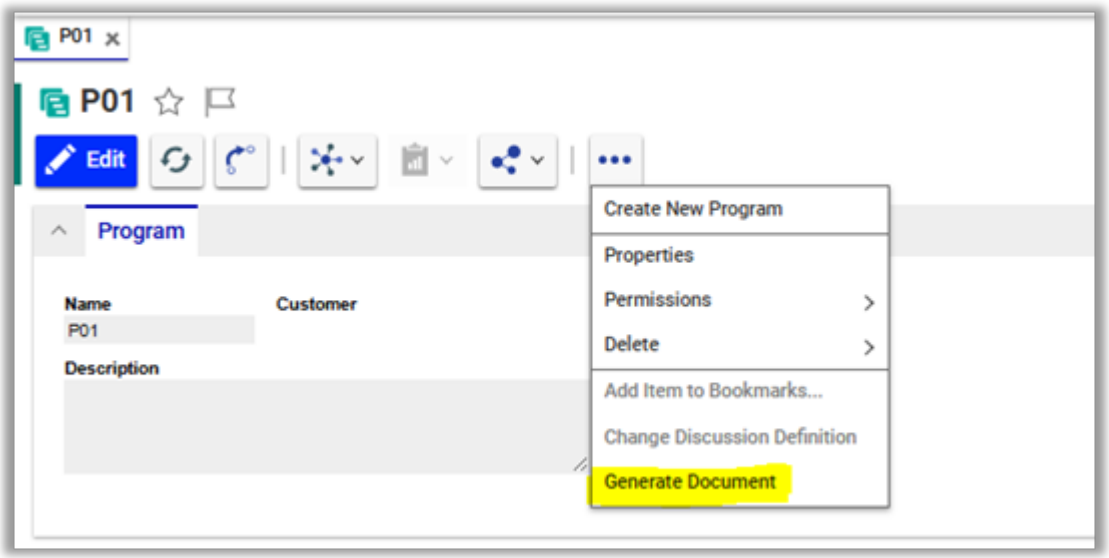


Configure Custom action for Word Generator

To generate Word or PDF documents using this generator, it can be configured as a custom action or added to CUI menus within Innovator. Below are the steps to use it via a custom action:

1. Create a custom action on the context item type.
2. Attach a client method that calls the server method “dg_WordDocument_GenerateDocument”. This server method internally utilizes the DocumentService from the Document Generator to process and generate the Word or PDF document.

3. The method “dg_WordDocument_GenerateDocument” can be executed on context item, which will be used to replace the properties values in Document template.
   e.g.



4. Additionally, a custom server method can be introduced to modify values on the context item before processing. Instead of directly invoking the 'dg_WordDocument_GenerateDocument' method, you can create a separate method that sets the required values on the context item. After that, the DocumentService object can be instantiated and the relevant APIs can be called.

   e.g.

```
Item contextItem = inn.newItem(itemType, "get");
contextItem.setID(itemId);
contextItem = contextItem.apply();
```

```
 Prorigo.Plm.DocumentGenerator.IDocumentService documentService = new
Prorigo.Plm.DocumentGenerator.Aras.DocumentService(inn,CCO);
 var fileItem = inn.newItem("File");
 var fileId
=documentService.GenerateWordDocument(contextItem,"GetDocumentAssociationJso
//Boolean parameter to indicate file to be checked in to item as per
configuration or not
 fileItem.setID(fileId);
 var resultItem = fileItem.apply("get");
 return resultItem;
```

ProAppDesigner Rule Engine - User Guide

Introduction

Purpose

This User Guide describes how to use the ProAppDesigner Rule Engine for external users.

Scope

This document provides instructions for configurations and functionality of Rule Engine application.

Target Audience

This document is intended for external users with limited access to data.

Guide Organization

This document describes how to use the ProAppDesigner Rule Engine application. This User Guide is organized into the following sections:

SECTION 1: **Introduction**	This section outlines the purpose and scope of this User Guide, as well as its intended target audience. It supplies a brief overview of each section to help users in navigating the guide.
SECTION 2: **Introduction to Rule Engine**	This section outlines the purpose of Rule Engine, it supplies an overview of the functionality provided in the Rule Engine. It also includes description for Creation of rule.

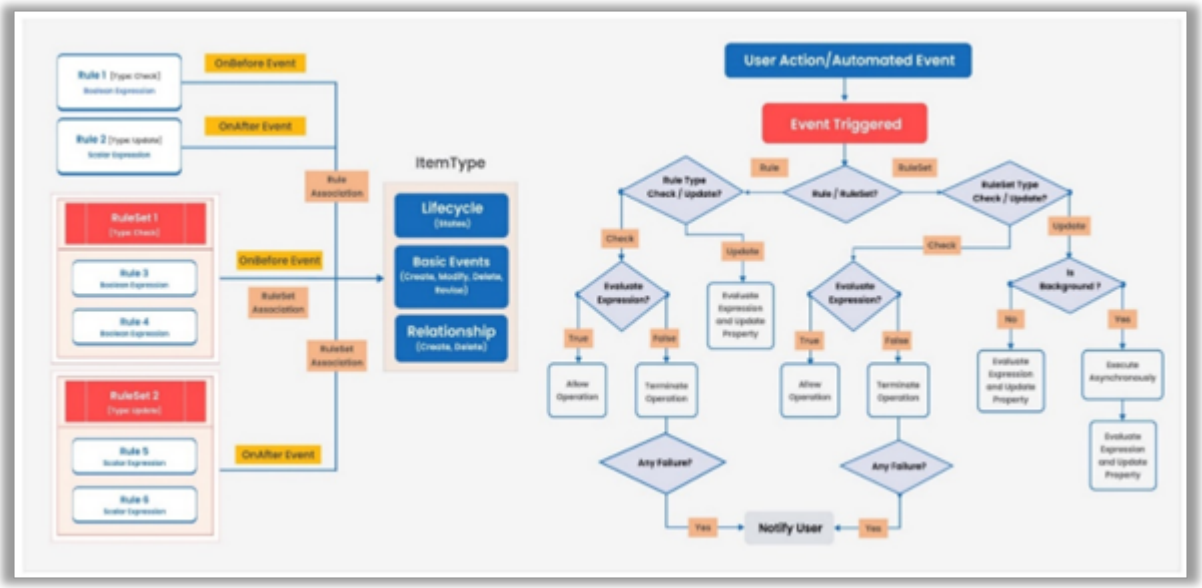
Introduction to ProAppDesigner Rule Engine

ProAppDesigner Rule Engine is a No-Code Platform that enables business users to define logic within rules, which are triggered by specific server events.

Below are key elements:

1. Rule
2. Rule Validation/Simulation
3. RuleSet
4. Associating Rule/RuleSet with ItemType Events
5. Rule Execution

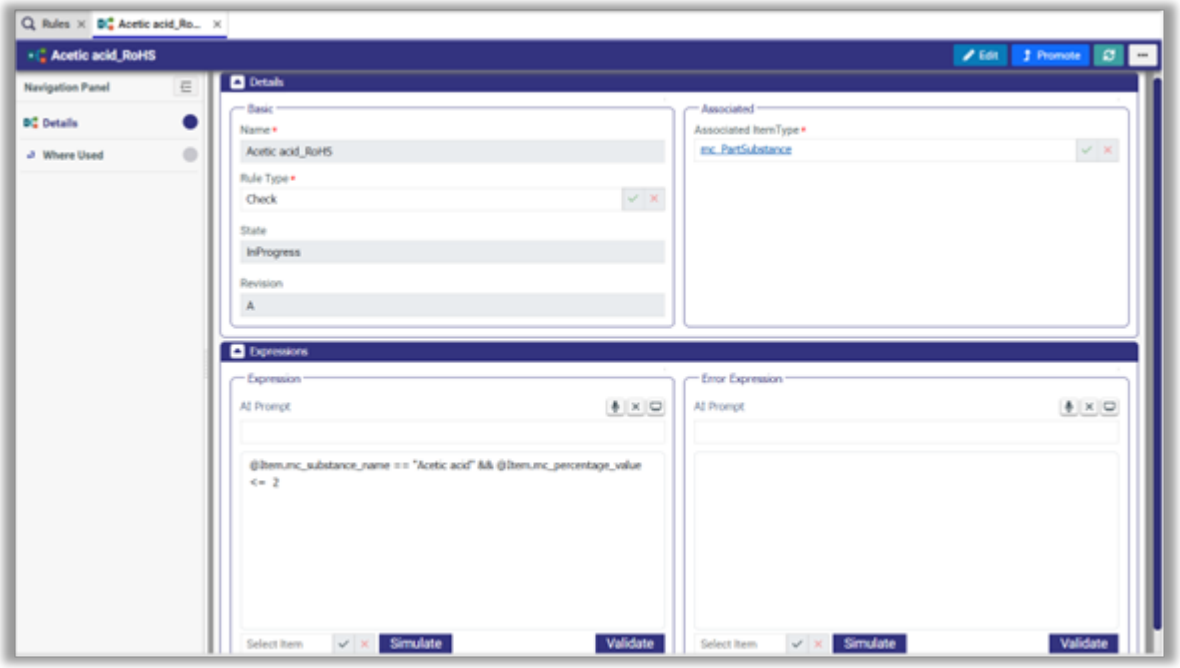
Schematic Diagram



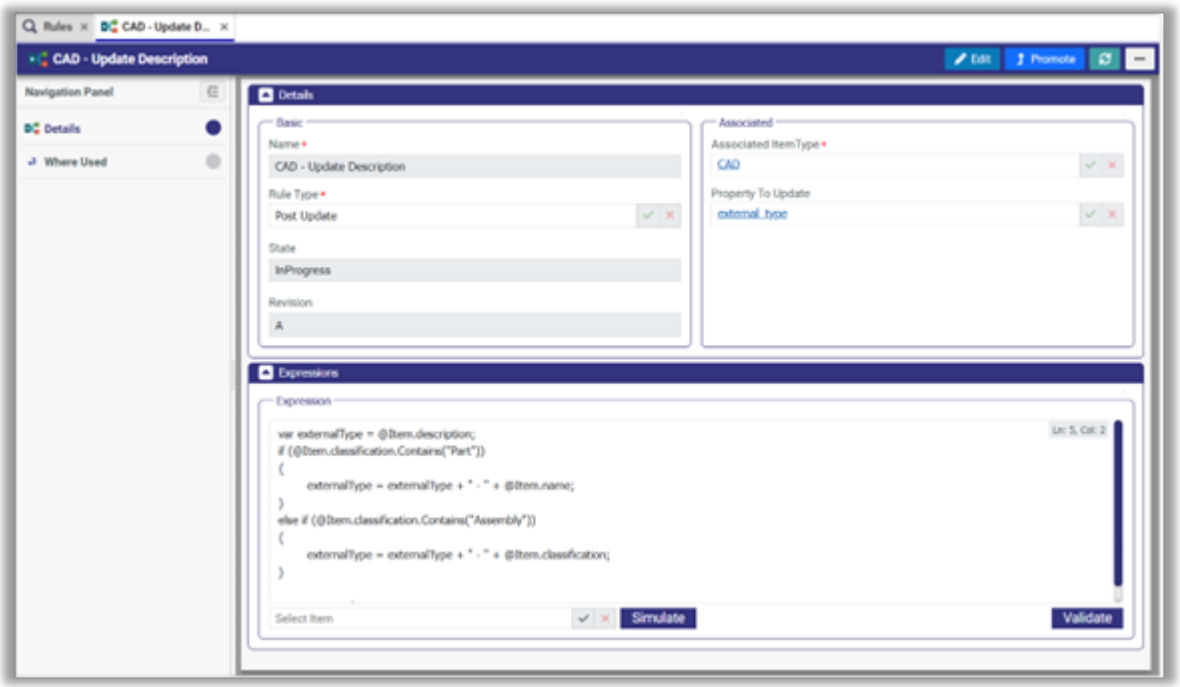
Rule

1. Associated ItemType: Name of ItemType for which this rule is defined
2. Rule Type
    - Check – To perform any kind of validations before any action. {create/edit/delete/revision/promote/relationship(add/delete)}
    - Pre-Update – To evaluate the rule expression and update the context item with the result of rule expression.
    - Post Update – To update the properties of the context item with the result of rule execution.
3. Property to Update
    - In case of Pre/Post Update rule type, select the property of associated ItemType which needs to be updated.
4. Expression
    - Capability to write any C# compatible code inside the expression. It can be any complex code or a mathematical expression.
    - Supports Item Properties, User Properties & Aras Server Methods.
5. Error Expression
    - In case of Check rule type, the error message to be shown to user. This can be an expression itself.

6. Check Rule

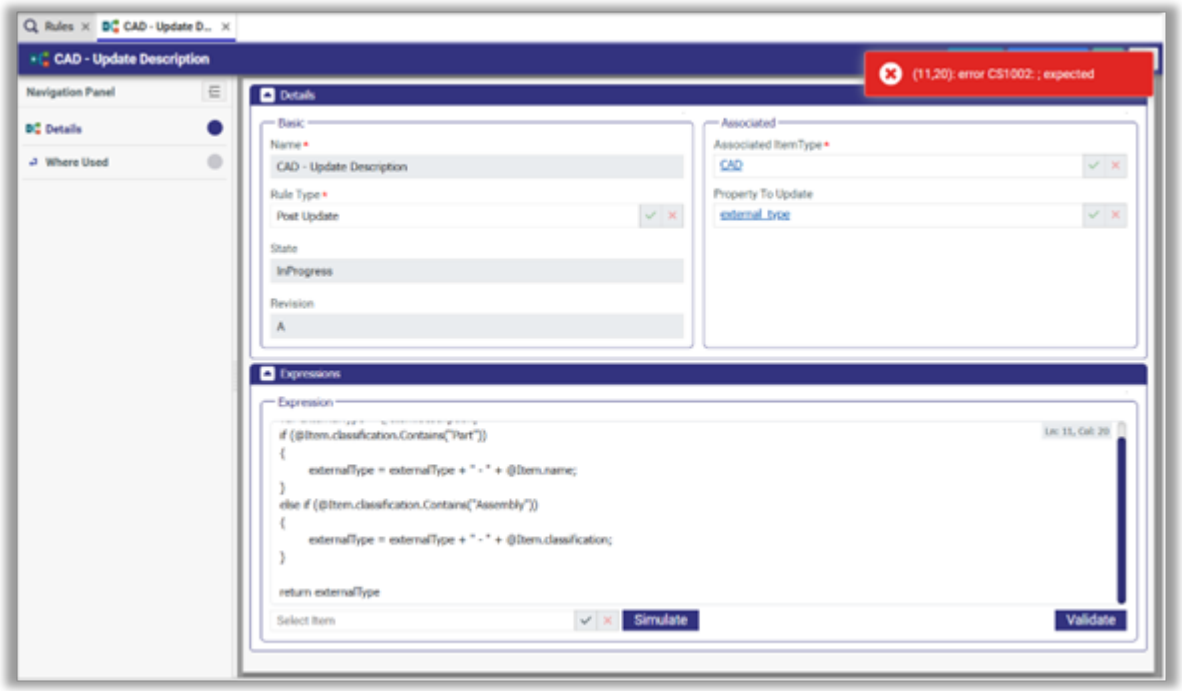


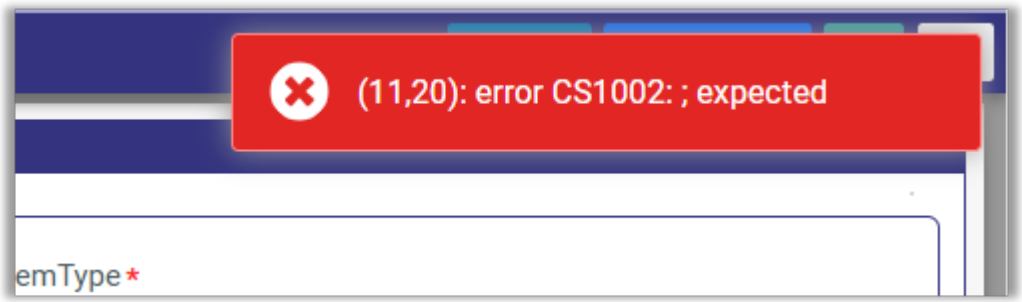
7. Update Rule



Rule Validation

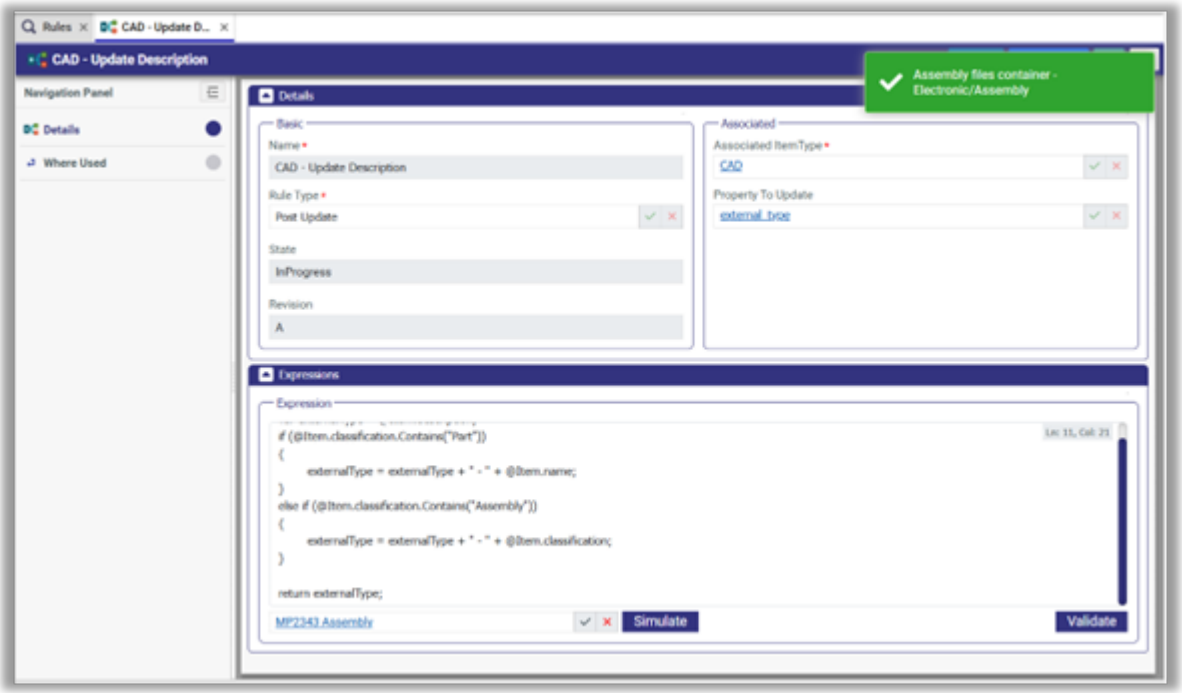
- Rule Validation is available to user to check if expression is correct. It will check for any syntax errors, property and method validation.
- If there are any errors, user will be presented with error message.





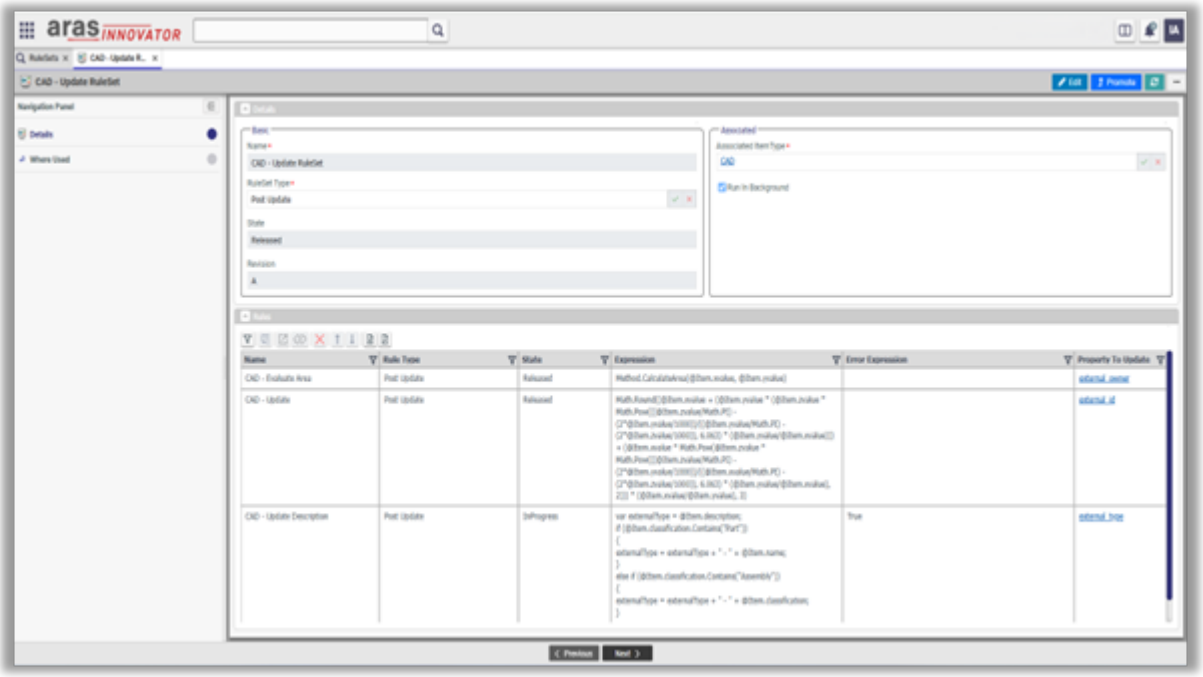
Rule Simulation

Rule Simulation is available to user to check if expression is returning desired value.



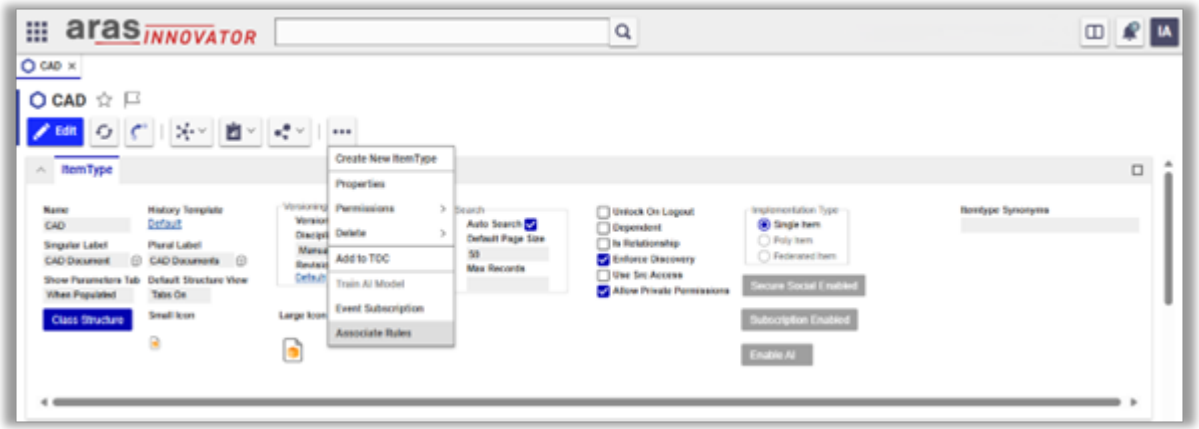
RuleSet

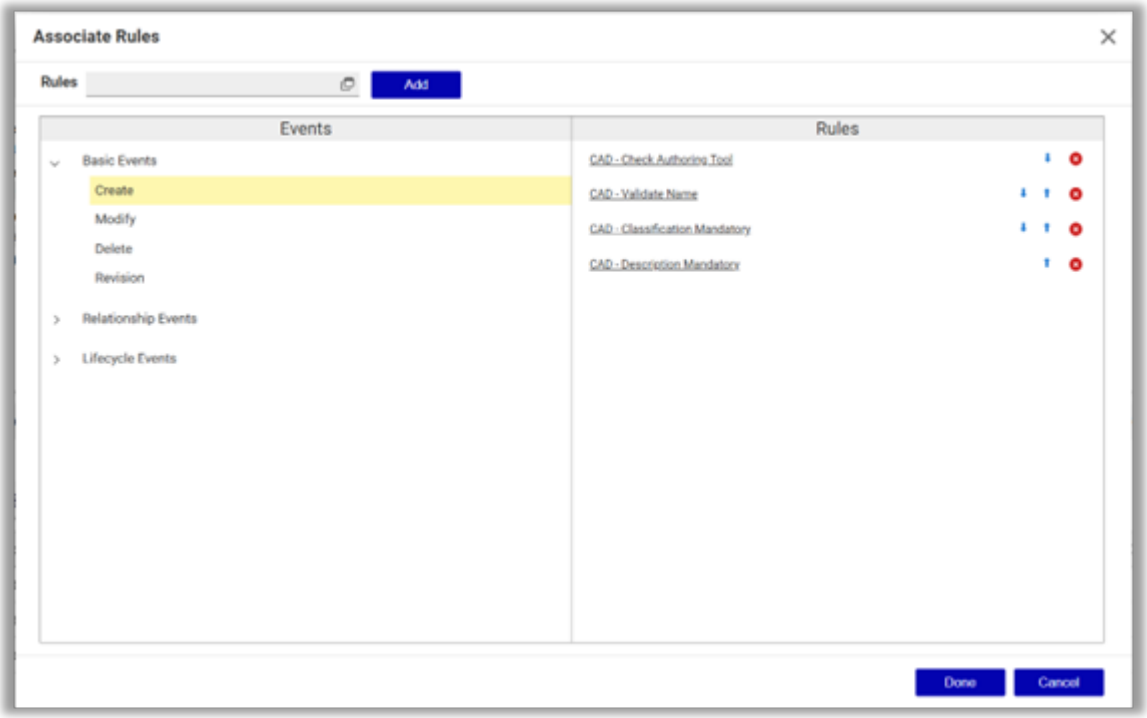
1. It is collection of Rules
2. Associated ItemType: Name of ItemType for which this ruleset is defined
3. Rule Type
    - Check – To perform any kind of validations before any action {create/edit/delete/revision/promote/relationship(add/delete)}
    - Post Update – To update the properties of the context item with the result of rule execution.
4. Run In Background
    - In case of Post Update ruleset type, execution of rules will be done asynchronously using background job. User will be notified upon completion of job.



Associate Rule/RuleSet

- “Associate Rules” action is available on ItemType
- User can select Rules/RuleSets which are defined for that particular itemtype and are in Released state



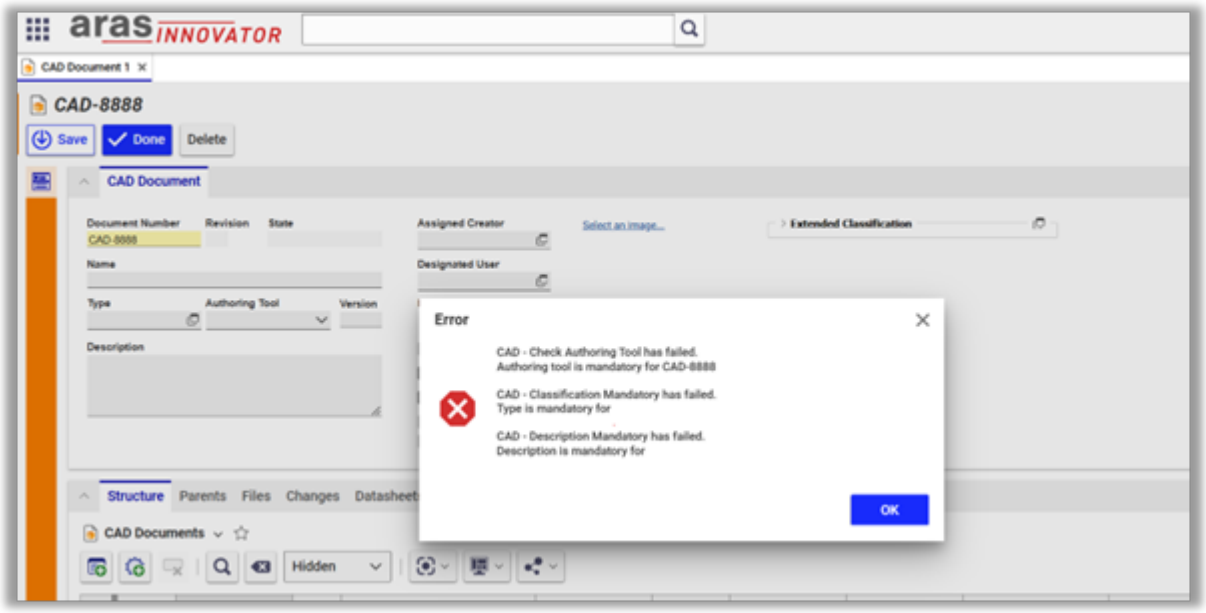


- Available Events
  - Basic Events

        - Create
        - Modify
        - Delete
        - Revision
    - Lifecycle Events – Promotion on Lifecycle state
    - Relationship Events – Add/Delete of relationship items

Rule Execution

- Once the rules/rulesets are associated with particular server events, framework takes care of executing those rules on those server events.
- In case of failure user will be notified.



Aras ProApp Designer - Developer Guide

Introduction

Purpose

This User Guide describes how to use the ProAppDesigner API’s for developers.

Scope

This document provides information about APIs supported in ProAppDesigner application.

Target Audience

This document is designed for developers seeking to customize the template using JavaScript.

Guide Organization

This guide explains how to integrate and utilize the ProAppDesigner APIs within your application. This Developer Guide is organized into the following sections:

SECTION 1: **Introduction**	This section outlines the purpose and scope of this Developer Guide, as well as its intended target audience. It supplies a brief overview of each section to help users in navigating the guide.
SECTION 2: **ProAppDesigner API Overview**	This section defines the purpose of the ProAppDesigner API.
SECTION 3: **Events**	This section details the range of events that can be triggered from the Wizard and Wizard Controls.
SECTION 4: **Event Handling**	This section covers handling events using custom scripting.
SECTION 5: **Wizard and Wizard Control API**	This section provides an overview of the various Wizard Control APIs available in ProAppDesigner.
SECTION 6: **Messages**	This section outlines the different Message APIs and their functions.
SECTION 7: **Progress Ring**	This section outlines the Progress Ring APIs and their usage.
SECTION 8: **ApplyMethod**	This section outlines various server methods and how to invoke them from the client side.
SECTION 9: **Examples**	This section provides examples of how to set and retrieve data from various UI controls.

ProAppDesigner API Overview

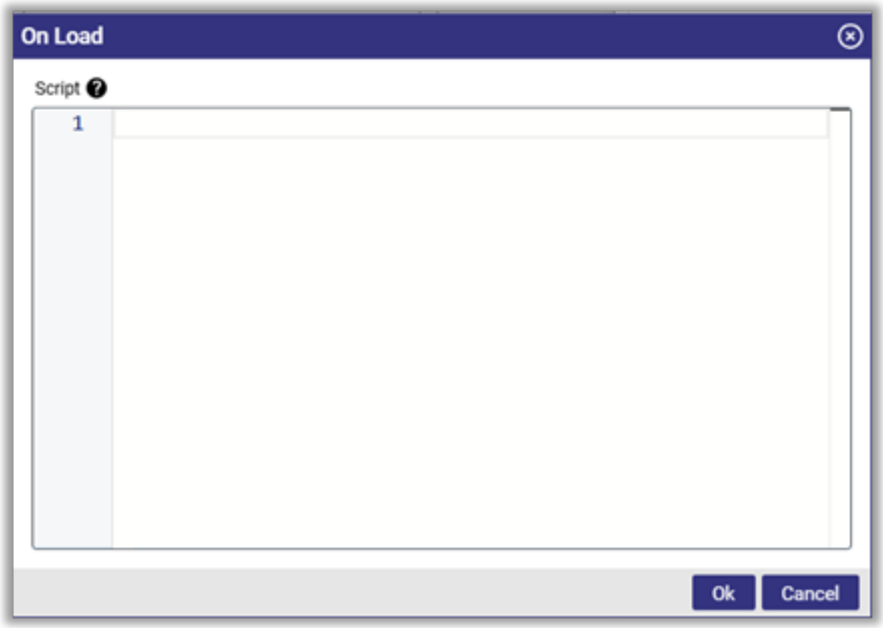
Aras ProAppDesigner Developer Guide is a comprehensive resource for customizing templates created using ProAppDesigner.

ProAppDesigner is a no-code solution for building user interfaces in Aras Innovator. Most of the UI can be created through configuration in TemplateStudio, using a rich set of controls available in the Toolbox. However, certain use cases may require writing code to implement custom logic.

The Developer Guide explains the events and JavaScript APIs needed to interact with the controls used in templates. It also provides APIs for reading data from and writing data to a control, provided the control's specific details are known.

Events

ProAppDesigner enables customization through a variety of event types that are triggered by the Wizard at different points during runtime. While configuring a ProAppDesigner template, administrators can register custom scripts for any of the supported events. These events are available on various controls within TemplateStudio, and custom scripts can be associated with them through the control's settings.



OnClick Event

The OnClick event is exclusively supported on the Button control. This event is triggered whenever the button is clicked, allowing you to execute custom logic in response to the user interaction.

// on button click opening custom flyout

```
appStudioWizard.showDialogFlyout(pageDialogName);
```

OnLoad Event

The OnLoad event is triggered after the data has been loaded for either the Wizard page or individual collection controls. This allows you to execute custom logic once the data is available, enabling dynamic behavior or additional initialization based on the loaded content.

//Setting list options on a column of the Table control:

```
const itemId = appStudioWizard.getItemId();
```

*// calling a server method & passing the **itemId** to get the required options. Refer to **2.7 Examples** on how to call server method.*

// building options map

```
let options = {
optionValue: "OptionLabel",
optionValue2: "OptionLabel2"
};
appStudioWizard.setCollectionListOptions("table_name", "column_name", options);
```

OnChange Event

The OnChange event is triggered whenever the value of a control is changed. It is supported on both simple controls (like text boxes, checkboxes, etc.) and columns within collection controls (such as tables or worksheets).

For **collection controls**, when the **OnChange** event is triggered, the eventData object includes both the **selectedItem** and the **columnPropertyName**. These allow you to access the value that was changed.

// On value change of control1, access its value and set the value of control2

```
let control1Value = appStudioWizard.getControlValue("control1");
if (control1Value) {
let value = control1Value === "Value1" ? "Discoverable" : "Hidden";
appStudioWizard.setControlValue("control2", value);
}
```

OnBeforeLoad Event

The OnBeforeLoad event is triggered before data is loaded into collection controls on a page. This event is supported only on table, worksheet, treeTable controls. OnBeforeLoad event can be used to set filterExpression on the collection control to load only required dataset.

//Setting filterExpression on Table control
// creating an expression string

```
let expressionString =
"classification EqualTo Item AND name StartsWith AS_";
```

*// Refer to section **filterCollectionItems** on how to build your expression string*
// calling filter API for table control

```
appStudioWizard.filterCollectionItems(controlName, expressionString);
```

OnSelect Event

The OnSelect event is triggered when an item within a collection control is selected by the user.

// getting the selected item to access its data

```
const selectedControlItems = appStudioWizard.getControlSelectedItems(
"supplierParts");
if (selectedItems) {
const selectedItem = selectedItems[0];
```

// accessing data of selected item

```
 const itemId = selectedItem.getItemId();
const relatedId = selectedItem.getRelatedId();
const partWeight = selectedItem.getValue("weight");
}
```

OnSubmit Event

The OnSubmit event is triggered after the Submit button is clicked within a flyout. This event is supported exclusively for flyout.

// accessing form values in the flyout & saving them

```
let price = appStudioWizard.getControlValue("price");
let date = appStudioWizard.getControlValue("purchased on");
let managedBy = appStudioWizard.getControlValue("managed_by_id");
```

*// calling a server method to save the data. Refer to **2.7 Examples** on how to call server method.*

```
if (onSuccess) {
appStudioWizard.showSuccessMessage("Item saved successfully");
} else {
appStudioWizard.showErrorMessage("Failed to save the Item.");
}
```

OnInsert Event

The **OnInsert** event is triggered after an item has been inserted into a collection control. The event handler receives the details of the inserted item through the eventData object, which is available within the scope of the event handler code.

```
// Accessing data of inserted item


const insertedItemCount = eventData.length;
for (let i = 0; i < insertedItemCount; i++) {

  // accessing data of selected item

  const relationshipId = eventData[i].getItemId();
const relatedItemId = eventData[i].getRelatedId();
const partWeight = eventData[i].getValue("weight");


    // setting a cell value on inserted item

  eventData[i].setValue("classification", "item");
}
```

OnSelectFlyout Event

The **OnSelectFlyout** event is triggered when opening the SelectItem flyout and is supported by the Item and Items controls. In this event handler, the admin can implement custom scripts and use the `setSelectFilterExpression API` to apply a custom filter expression.

```
// creating a custom expression string
let expressionString = "classification EqualTo Item AND id NotEqualTo 8D58C2D8DC1649279E64CCF5BD2FF7A7";
// Refer to filterCollectionItems section on how to build your expression string
appStudioWizard.setSelectFilterExpression(controlName, expressionString, enforce);
```

OnInsertFlyout Event

The **OnInsertFlyout** event is triggered when opening the InsertItem flyout and is supported in Table, Worksheet, and TreeTable controls. In the event handler, the admin can write custom scripts and use the setInsertFilterExpression API to apply a custom filter expression.

// creating a custom expression string

```
let expressionString = "classification EqualTo Item AND id NotEqualTo 8D58C2D8DC1649279E64CCF5BD2FF7A7";
```

*// Refer to **filterCollectionItems** section on how to build your expression string*

```
appStudioWizard.setInsertFilterExpression(controlName, expressionString, enforce);
```

OnSave Event

The OnSave event is triggered after saving the item within the wizard or in section/group controls that have their own associated items.

```
const itemId = appStudioWizard.getItemId();
const itemType = appStudioWizard.getItemType();
```

*// calling a server method to save custom data. Refer to **2.7 Examples** on how to call server method.*

```
if (onSuccess)
appStudioWizard.showSuccessMessage("Data saved successfully");
else
appStudioWizard.showErrorMessage("Failed to save the custom data.");
```

OnBeforeSave Event

The OnBeforeSave event is triggered before the actual save request is sent to the server. Within its event handler, you can implement code to validate changes and cancel the save operation by returning false if any issues are detected.

```
var isItemNew = appStudioWizard.isItemNew();
var controlValue = appStudioWizard.getControlValue('controlName');
```

// Before calling a server method to save data. Refer to ApplyMethod Examples on how to stop server method to save data.

```
if (isNew && !controlValue)
true;
else
appStudioWizard.showErrorMessage("Failed to save the custom data.");
false
```

Event Handling

A custom script can be defined as an event handler using the predefined variable appStudioWizard. This script can include any JavaScript code that reads values from wizard controls, makes requests to the server using the Innovator IOM API, and updates controls based on the server's response. It can also make requests to third-party servers via JavaScript HTTP clients. ProAppDesigner Wizard provides a simple API accessible through appStudioWizard to interact with different control types. Additionally, the wizard offers APIs to display various message types to users during event handling. If the event handler's code takes time to execute, there is also an API to show a progress ring to indicate ongoing processing.

Wizard and Wizard Controls API

Wizard controls are categorized into two types: simple controls and collection controls. For simple controls, you can easily get or set values using the getControlValue and setControlValue APIs. For collection controls, you first need to retrieve the selected items by calling the getControlSelectedItems API. Each selected item is represented by an ItemControl, which allows you to get or set column values using the getValue and setValue APIs.

To set or get values from controls, you can pass the control name to the API methods. However, if there are multiple controls with the same name, you need to use a qualified path instead of just the control name to uniquely identify the specific control when calling the API methods.

QualifiedPath for controls used in the group control:

pageName/sectionName/groupName/controlName
QualifiedPath for controls used in the panel control:
pageName/sectionName/groupName/panelControlName/pageName/sectionName/groupName/controlName

Example:
appStudioWizard.getControlValue("pageName/sectionName/groupName/controlName");

Wizard

The Wizard object can be accessed in scripts using the keyword AppStudioWizard, which provides all the APIs needed to interact with the various controls within the Wizard.

getItemId

Get itemId of the item loaded in the Wizard.

```
var itemId = appStudioWizard.getItemId()
```

getItemType

Get itemType of the item loaded in the Wizard.

```
var itemTypeName = appStudioWizard.getItemType()
```

getItemConfigId

Get itemConfigId of the item loaded in the Wizard.

```
var itemConfigId = appStudioWizard.getItemConfigId()
```

isItemNew

Returns true if the item in the Wizard is never saved to server.

```
var isItemNew = appStudioWizard.isItemNew()
```

isItemEditable

Returns true if the item in the Wizard is editable (in edit mode).

```
var isItemEditable = appStudioWizard.isItemEditable()
```

Section and Group controls on a page can have their own ItemType context. The following API returns true if the associated item of the layout control (Section or Group) is editable (i.e., the page is in edit mode).

```
var isLayoutEditable = appStudioWizard.isItemEditable(layoutControlName)
```

isControlUpdated

Checks control with the given name is updated (has unsaved changes)

```
var isControlUpdated = appStudioWizard.isControlUpdated(controlName)
```

reload

Reload the entire wizard with the latest item details from the server, discarding all local changes.

```
appStudioWizard.reload();
```

reloadControl

Reload a specific collection control's data from the server, discarding all local changes.

```
appStudioWizard.reloadControl(controlName);
```

showDialogFlyout

Shows the flyout given the flyout name.

```
appStudioWizard.showDialogFlyout(pageDialogName);
```

showInsertFlyout

Shows the associated InsertItem flyout on the collection controls

```
appStudioWizard.showInsertFlyout(controlName);
```

showWorkflowActivityFlyout

Shows WorkflowActivity flyout given the activityId, workflowId, itemId and tableControlName.

```
appStudioWizard.showWorkflowActivityFlyout(activityId, workflowId,
itemId, tableControlName);
```

downloadFile

Downloads a file from vault server given the fileId and fileName.

```
appStudioWizard.downloadFile(fileId, fileName);
```

Aras Object

Aras object can be accessed in scripts using the keyword Aras. It provides APIs to interact with Innovator on the client side. However, it is only available for templateViews used to display the Wizard within the Innovator UI and does not function when the templateView is used to show the UI in the Portal.

Simple Controls

Following API works with all simple controls like Text, List, Checkbox, RichText, Item and Items.

getControlValue

Gets the value of the simple control based on its name.

```
var controlValue = appStudioWizard.getControlValue(controlName);
```

setControlValue

Sets value for the simple control based on its name.

```
appStudioWizard.setControlValue(controlName, controlValue);
```

setSelectFilterExpression

This API sets a default filter expression inside the SelectItem flyout when it opens for selecting an item. It is supported for Item and Items controls. Additionally, an optional boolean parameter, enforce, can be passed to hide the configured filter expression from the user.

// creating a custom expression string

```
let expressionString = "classification EqualTo Item AND id NotEqualTo
8D58C2D8DC1649279E64CCF5BD2FF7A7";
```

*// Refer to **filterCollectionItems** section on how to build your expression string*

```
appStudioWizard.setSelectFilterExpression(controlName, expressionString,
enforce);
```

setControlLabel

Control label can be updated, given its control name & label value.

```
appStudioWizard.setControlLabel(controlName, label);
```

Collection Controls

The following API functions with collection controls such as Table, Worksheet, TreeTable, Structure, DashboardTable, and DashboardWorksheet. These collection control APIs can only be used when the controls are visible on the current page.

getControlSelectedItems

Gets list of selected items from a collection control given the collection control name.

```
var controlItems =
appStudioWizard.getControlSelectedItems(collectionControlName);
```

setControlSelectedItems

This API selects items in a collection control by specifying the collection control name along with an array of relationship IDs when the table displays relationship data. For dashboard tables, dashboard worksheets, and reference items tables, you can pass item IDs directly instead.

```
const arrayOfIds = ["131706F6B80D47A5A68ED00BAF210CC8"]
appStudioWizard.setControlSelectedItems(collectionControlName,
arrayOfIds);
```

getControlItem

Retrieves an item from a collection control by specifying the control name and either the relationshipId or itemId. Use relationshipId when the table represents relationships, and itemId when the table represents dashboard data or references.

```
var itemControl = appStudioWizard.getControlItem(collectionControlName,
relationshipId/itemId);
```

getControlItems

Retrieves all items from a collection control by specifying the control name. This method returns all items currently bound to the control.

```
var items = appStudioWizard.getControlItems(collectionControlName);
```

getControlItemByRelatedId

Retrieves an item from a collection control using the relatedItemId and the control name. It works only for collection controls that display non-null relationship data.

```
var controlItems = appStudioWizard.getControlItemByRelatedId(collectionControlName, relatedItemId);
```

addControlItem

This API adds a new item to a collection control—such as a table, worksheet, or tree table—that displays relationship data. The new item is inserted immediately after the specified relationshipId within the collection. The function requires the collection control name and an ItemData object containing all relevant properties and their values. Upon successful insertion, the newly added item is returned.

```
const itemData = {
properties: {
address: "Sample Data",
},
relatedItem: {
properties: {
name: "Sample Item",
description: "Sample Desc",
},
},
};
```

```
var addedControlItem = appStudioWizard.addControlItem(collectionControlName, relationshipId, itemData);
```

addControlItems

Adds multiple items to a dynamic table control by passing an array of itemData objects, each containing all necessary properties and values. This method only supports dynamic table controls

that display relationship data and returns an array of all inserted items.

```
const itemDataArray = [
{
properties: {
name: 'Sample Item 1',
description: 'Sample Desc 1',
},
},
{
properties: {
name: 'Sample Item 2',
description: 'Sample Desc 2',
},
},
]
var addedControlItem =
appStudioWizard.addedControlItems(collectionControlName, itemDataArray);
```

insertControlItem

This API inserts an item into a collection control—specifically a table, tree table, or worksheet that displays relationship data—immediately after the specified relationshipId. It requires the collection control name and returns the inserted item.

```
 appStudioWizard.insertControlItem(collectionControlName, relationshipId,
insertItemId).then((insertedItemControl) => {
// accessing inserted ItemControl and setting its value
insertedItemControl.setValue(columnPropertyName, cellValue);
});
```

deleteControlItem

This API deletes an item from a collection control. For relationship tables, you provide the collection control name and the relationshipId of the item to delete. For tables displaying reference items or

items, you pass the itemId instead.

```
appStudioWizard.deleteControlItem(collectionControlName,
relationshipId/itemId);
```

setCommandDropdownOptions

Each collection control includes its own toolbar with various commands. If a command in the toolbar is a dropdown, you can set the options for that dropdown by using this function, providing the collection control name and the command name.

```
var options = [
{ label: 'optionLabel', value: 'optionValue' },
{ label: 'optionLabel2', value: 'optionValue2' },
];
```

```
appStudioWizard.setCommandDropdownOptions(collectionControlName,
commandName, options);
```

filterCollectionItems

This API is applicable only to Table, Worksheet, and Report controls that support expression-based filtering. By invoking this function and passing an expressionString, the items displayed in the control will be filtered accordingly. The filtering is executed on the server, and the control's data is refreshed to reflect the updated results.

You can use this API in the OnBeforeLoad event of the control to load data dynamically based on the specified filter expression. Alternatively, it can be used in a custom toolbar command's onClick handler. However, when used this way, you must also call reloadControl immediately afterward to ensure the data in the control is refreshed according to the new filter. For Report controls, this API is only applicable if the report was created using an expression-based query.

```
appStudioWizard.filterCollectionItems(collectionControlName,
expressionString);
```

Simple Expression: A simple expression consists of a property name, an operator, and a value. The type of operator that can be used in the expression depends on the data type of the selected property. To view the list of supported operators for a specific property, you can access the Property Filter section in the Properties flyout of the respective control.

" `propertyName OPERATOR value` "

Following operators can be used to create an expression based of the property's data type.

Contains, EqualTo, NotEqualTo, LessThan, GreaterThan, LessThanEqualTo, GreaterThanEqualTo, StartsWith, EndsWith, DoesNotContains, DoesNotStartsWith & DoesNotEndsWith.

Complex Expression: Complex expression is composed of many simple expressions with logical and grouping operators.

" `expression LOGICAL_OPERATOR expression LOGICAL_OPERATOR expression` "

Logical operators 'AND' and 'OR' can be used along with grouping operators '(' and ')' to a create complex expressions that contain simple expressions.

Sample Expression:

```
 const expressionString = "id NotEqualTo
131706F6B80D47A5A68ED00BAF210CC8"
 const expressionString = "name Contains app OR name EndsWith item"
 const expressionString = "classification EqualTo Item AND name StartsWith
AS_"
```

setInsertFilterExpression

This API allows you to set a **default filter expression** inside the **InsertItem flyout** when it is opened to select an item for insertion. It is supported in **Table**, **Worksheet**, and **TreeTable** controls. You can optionally pass a **boolean parameter enforce**, which, if set to true, will hide the configured filter expression from the user.

// creating a custom expression string

```
let expressionString = "classification EqualTo Item AND id NotEqualTo 8D58C2D8DC1649279E64CCF5BD2FF7A7";
```

*// Refer to **filterCollectionItems** section on how to build your expression string*

```
appStudioWizard.setInsertFilterExpression(controlName, expressionString, enforce, typeName);
```

setControlParameters

This API can be used on Structure, Graph, and Report controls to pass custom parameters to the underlying QueryDefinition or ServerMethod. It accepts a JavaScript object containing key-value pairs that represent the parameters to be passed. These parameters will be sent to the server and used during data retrieval. You can use this API in the OnBeforeLoad event of the control to ensure the data is loaded based on the supplied custom parameters. Alternatively, it can be used in the onClick handler of a custom toolbar command. In that case, you must call the reloadControl API immediately after, to refresh the control's data using the updated parameters.

```
appStudioWizard.setControlParameters(controlName, {classification:"Component"});
```

getStructureActiveTab

Gets the typeName of current active tab of structure control given the structure control name and selected node itemId.

```
appStudioWizard.getStructureActiveTab(controlName, itemId);
```

setStructureActiveTab

Sets the currently active tab in the structure control based on the structure control name, the selected node's itemId, and the tab's typeName. For regular relationships, the typeName is a combination of the relationship name and the related ItemType name, as illustrated in the example below. In the case of null relationships, the typeName consists of only the relationship name. When the tab

displays an item from an ItemProperty or a reference item, the typeName should be the ItemType name alone.

```
const typeName = “relationshipName(relatedItemTypeName)”;
appStudioWizard.setStructureActiveTab(controlName, itemId, typeName);
```

Control Item

A control item represents an item displayed in the collection control, either at the root level or as a child. It provides an API to set and retrieve property values, as well as to add or remove child items from the parent control item.

getItemId

Once you have controlItem that represents a row in collection controls, associated relationshipId/itemId can be obtained by calling this function.

```
var itemId = controlItem.getItemId();
```

getRelatedId

Once you have controlItem that represents a row in collection controls, associated relatedId can be obtained by calling this function.

```
var relationshipId = controlItem.getRelatedId();
```

isItemNew

Once you have controlItem that represents a row in collection controls, associated isNew state can be obtained by calling this function. Return true if the item is newly added or inserted.

```
var isItemNew = controlItem.isItemNew()
```

getValue

Once you have a controlItem representing a row in a collection control, you can retrieve the value of a specific cell by providing the column's property name. If the controlItem represents relationship data, and you need to access a property of the related item, the columnPropertyName should be specified in the format "related.relatedItemPropertyName".

```
var cellValue = controlItem.getValue(columnPropertyName);
```

setValue

Once you have itemControl that represents a row in collection controls, a particular cell value in that row can be set by passing column property name and cell value. If the controlItem represents relationship data, to set relatedItem property value, you need to pass columnPropertyName as “related.relatedItemPropertyName”.

```
itemControl.setValue(columnPropertyName, cellValue);
```

getChildItems

Once you have controlItem that represents a row in worksheet, treeTable, structure, graph controls, its child items can be retrieved by passed child relationshipName.

```
var childControlItems =
controlItem.getChildItems(childRelationshipName);
```

getChildItem

Once you have controlItem that represents a row in worksheet, treeTable, structure, graph controls, its child item can be retrieved by passing child relationshipName and child relationshipId

```
var childControlItem = controlItem.getChildItem(childRelationshipName,
childRelationshipId);
```

getChildItemByRelatedId

Once you have controlItem that represents a row in collection controls, childControlItem can be obtained given that child relationshipName, child relatedId.

```
var childControlItems = controlItem.getChildItemByRelatedId(
childRelationshipName, childRelatedItemId);
```

addChildItem

Once you have controlItem that represents a row in collection controls, childItem can be added given that child relationshipName, child relationshipId, and itemData. It also returns newly added item in the form of childControlItem.

```
const itemData = {
properties: {
address: "Sample Data",
},
relatedItem: {
properties: {
name: "Sample Item",
description: "Sample Desc",
},
},
};
```

```
var childControlItem = controlItem.addChildItem(childRelationshipName, childRelationshipId, itemData);
```

insertChildItem

Once you have controlItem that represents a row in collection controls, childControlItem can be inserted given that child relationshipName, child relationshipId, and insertItemId. It also returns newly added item in the form of childControlItem.

```
controlItem.insertChildItem(childRelationshipName, childRelationshipId, insertItemId).then((insertedChildItemControl) => {
// accessing inserted ItemControl and setting its value
insertedChildItemControl.setValue(columnPropertyName, cellValue);
});
```

deleteChildItem

Once you have controlItem that represents a row in collection controls, childItem can be deleted from the controlItem given childRelationshiName and childRelationshipId.

```
controlItem.deleteChildItem(childRelationshipName, childRelationshipId);
```

getParentItem

Once you have controlItem that represents a node in TreeTable or Structure controls, parent item can be obtained by using this API.

```
var parentControlIttem = controlItem.getParentItem();
```

getItemType

Once you have controlItem that represents a row or node in collection controls, its relationship name can be obtained by calling this function. If the controlItem represent item from ItemProperty or reference item, then you will obtain its ItemType name.

```
var itemTypeName = controlItem.getItemType();
```

getItemConfigId

Once you have controlItem that represents a row in collection controls, associated item configId can be obtained by calling this function.

```
var itemConfigId = controlItem.getItemConfigId();
```

getRelatedItemType

Once you have controlItem that represents a row or node in collection controls, its related ItemType name can be obtained by calling this function.

```
var relatedItemTypeName = controlItem.getRelatedItemType();
```

Layout Controls: Group & Section

Below APIs are supported for Group & Section controls when their Associated Type setting is set to **Type**.

setLayoutControlContext

Set itemId as context for a layout control based on its control name to refresh the control with passed itemId.

```
appStudioWizard.setLayoutControlContext(sectionControlName, itemId);
```

getLayoutControlContext

Get itemId associated with a layout control based on its name.

```
var itemId =
appStudioWizard.getLayoutControlContext(sectionControlName);
```

expandSection

By passing the section control name and a Boolean parameter as true or false, section control can be expanded or collapsed respectively.

```
appStudioWizard.expandSection(sectionControlName, true);
```

List Control

setListOptions

Set list options for a List control by its control name.

```
 const options = { optionValue: "OptionLabel", optionValue2:
"OptionLabel2" };
 appStudioWizard.setListOptions(listControlName, options);
```

setCollectionListOptions

Collection controls can have column that represents a list. Table, Worksheet, TreeTable, DashboardTable, DashboardWorksheet can have a column of type list. Following API sets list options for a collection control by passing its list column name.

```
 const options = {optionValue: "OptionLabel", optionValue2:
"OptionLabel2"};
 appStudioWizard.setCollectionListOptions(collectionControlName,
columnName, options);
```

setCollectionChildListOptions

Set list options for nested table for a collection control by passing its list column name and relationship name.

```
 const options = {optionValue: "OptionLabel", optionValue2:
"OptionLabel2"};
 appStudioWizard.setCollectionChildListOptions(collectionControlName,
relationshipName, columnName, options);
```

Frame Control

getElement

Gets DOM element of the control based on the control name. This is mainly used to get frame DOM element for the Frame control.

```
var htmlFrameElement = appStudioWizard.getElement(frameControlName);
if (htmlFrameElement) htmlFrameElement.src = "siteURL";
```

CADViewer Control

setControlValue

Sets cadDocumentId on the CADViewer control by its control name to load the cad model.

```
appStudioWizard.setControlValue(viewerControlName, "cadDocumentId");
```

DynamicTable Control

setTableConfiguration

Sets the configuration for a DynamicTable control by its control name. The configuration object defines the table definition, including its columns, data types, widths, labels (with localization support), and other display settings.

Each column object should specify a unique name, a localized label object, control definition (including data type and localized label), width (in pixels), visibility, and optionally a data source if the column represents a reference or item field.

The columnWidthByPercentage property determines whether widths are treated as fixed pixel values (false) or percentage-based (true).

```
const columns = [
{
name: 'supplier name',
label: 'Supplier Name',
control: {
name: 'supplier name',
label: 'Supplier Name',,
dataType: 'String',
},
width: 500,
hide: false,
},
{
name: 'team id',
label: 'Team',
control: {
name: 'team id',
label: 'Team',
dataType: 'Item',
```

```
 dataSource: 'Team',
 }
 width: 400,
 hide: false,
 }];
 configurationObject = {
 columns: columns,
 columnWidthByPercentage: false,
 }
   appStudioWizard.setTableConfiguration(tableControlName,
configurationObject);
```

Messages

The Message API allows you to display different types of messages, each with a corresponding background color. When any of these APIs is called, a toast notification appears in the top-right corner of the wizard and remains visible for 5 seconds.

showSuccessMessage

Show's success message as flash notification with green background.

appStudioWizard.showSuccessMessage(message);

showErrorMessage

Shows error message as flash notification with red background.
appStudioWizard.showErrorMessage(message);

showWarningMessage

Shows warning message as flash notification with orange background.

```
appStudioWizard.showWarningMessage(message);
```

showInfoMessage

Shows info message as flash notification with blue background.

```
appStudioWizard.showInfoMessage(message);
```

Progress Ring

The ProgressRing API is used to display a busy icon during time-consuming operations within a script. When the script makes a server call—whether through the IOM API or HTTPClient—the busy icon can be shown over the wizard to indicate ongoing activity.

showProgressRing

Shows busv icon on the Wizard.

```
appStudioWizard.showProgressRing()
```

hideProgressRing

Hides busv icon on the Wizard.

```
appStudioWizard.hideProgressRing()
```

ApplyMethod

It allows you to invoke any server method defined on the server side. All parameters required by the method must be packaged in JSON format when making the call.

Example 1: Calling serverMethod by passing parameters and getting item in the response

Client Code:

```
const parmObj = { partId: "FB3CBC878D5747DEBF25508B23CF6B5B" };
const resultObj = appStudioWizard.applyMethod("methodName", parmObj);
```

Server Method:

```
string partId = this.getProperty("partId");
if(partId == null) throw new Exception("Please provide partId.");

Item partItem = this.getInnovator().newItem("Part", "get");
partItem.setID(partId);
partItem = partItem.apply();
if(partItem.isError() && partItem.isEmpty()) throw new Exception("No
items found.");

return partItem;
```

Example 2: Calling serverMethod by passing json and getting json in the response

Client Code:

```
const partDetails = { classification: "Component", make_buy: "Make" };
const body = { request: JSON.stringify(partDetails) };
const resultObj = appStudioWizard.applyMethod("methodName", body);
```

Server Method:

```
string partDetailsString = this.getProperty("request");
var partDetails = new Newtonsoft.Json.Linq.JObject();
if (partDetailsString == null) {
throw new Exception("Please provide partDetails.");}
```

```
 else { partDetails =
Newtonsoft.Json.Linq.JObject.Parse(partDetailsString);}

 string classification =
partDetails.SelectToken("$.classification")?.ToString();
 string make buy = partDetails.SelectToken("$.make buy")?.ToString();
 Item query = this.getInnovator().newItem("Part", "get");
 query.setAttribute("select", "item_number, name, classification,
make buy");
 query.setProperty("classification", classification);
 query.setProperty("make buy", make_buy);
 var partItems = query.apply();
 if(partItems.isError() && partItems.isEmpty()) throw new Exception("No
items found");

 return partItems;
```

Output:

```
 [
 {
 "classification": "Component",
 "make buy": "Make",
 "name": "Trunk",
 "item number": "004",
 "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"
 },
 ...]
```

Examples

How to Set and Get Value on Checkbox Control

```
var controlName = "checkbox1";
appStudioWizard.setControlValue(controlName, false);
```

```
var controlValue = appStudioWizard.getControlValue(controlName);
console.log ("Control Value: " + controlValue);
```

//Control Value: false

How to Set and Get Value on Text Control

```
var controlName = "itemNumber";
appStudioWizard.setControlValue(controlName, "P00456");
```

```
var controlValue = appStudioWizard.getControlValue(controlName);
appStudioWizard.showSuccessMessage("Control Value: " + controlValue);
```

//Control Value: P00456

How to Set and Get Value on RichText Control

```
var controlName = "richtext1";
appStudioWizard.setControlValue(controlName, "<h2>Can set the value.</h2>");
```

```
var controlValue = appStudioWizard.getControlValue(controlName);
appStudioWizard.showSuccessMessage(controlValue);
```

//Control Value: `<h2>Can set the value.</h2>`

How to Set and Get Value for a Row on Table Control

```
var controlName = "table1";
var controlItems =
appStudioWizard.getControlSelectedItems(controlName);
controlItems[0].setValue("column2", "some value")
var cellValue = controlItems[0].getValue("column1");
appStudioWizard.showSuccessMessage(cellValue)
```

How to Configure DynamicTable Control

The DynamicTable is loaded at runtime through event handling, based on the provided configuration and data. You can define its structure in the OnBeforeLoad event, located within the Settings flyout in the Studio.

```
const tableObject = {
  columns: [
    {
      name: 'supplier_name',
      label: 'Supplier Name',
      control: {
        name: 'supplier_name',
        label: 'Supplier Name',
        dataType: 'String'
      },
      width: 500,
      hide: false
    },
    {
      name: 'team_id',
      label: 'Team',
      control: {
        name: 'team_id',
        label: 'Team',
        dataType: 'Item',
        dataSource: 'Team'
      },
      width: 400,
      hide: false
    },
    {
      name: 'effective_date',
      label: 'Effective Date',
      control: {
        name: 'effective_date',
        label: 'Effective Date',
        dataType: 'Date'
      },
      width: 300,
      hide: false
    },
    {
      name: 'contact_number',
      label: 'Contact Number',
      control: {
        name: 'contact_number',
        label: 'Contact Number',
        dataType: 'String'
      },
```

```
      width: 200,
      hide: false
    }
  ],
  columnWidthByPercentage: false
};

// Apply configuration to the table

appStudioWizard.setTableConfiguration('dynamicTable', tableObject);
```

During the OnLoad event, actual data can be loaded into the table. The following code snippet should be placed in the OnLoad event handler within the Settings flyout in the Studio.

```
const dataArray = [
  {
    id: 'SUPPLIER_ID_001',
    properties: {
      supplier_name: 'Supplier One',
      team_id: {
        data_source: { name: 'Team' },
        id: 'TEAM_ID_001',
        keyed_name: 'Engineering Team A'
      },
      effective_date: '2025-06-01T00:00:00',
      contact_number: '+1-555-100-1000'
    }
  },
  {
    id: 'SUPPLIER_ID_002',
    properties: {
      supplier_name: 'Supplier Two',
      team_id: {
        data_source: { name: 'Team' },
        id: 'TEAM_ID_002',
        keyed_name: 'Production Team B'
      },
      effective_date: '2025-06-03T00:00:00',
      contact_number: '+1-555-200-2000'
    }
  },
  {
    id: 'SUPPLIER_ID_003',
    properties: {
      supplier_name: 'Supplier Three',
      team_id: {
        data_source: { name: 'Team' },
        id: 'TEAM_ID_003',
        keyed_name: 'Quality Team C'
      },
      effective_date: '2025-06-05T00:00:00',
      contact_number: '+1-555-300-3000'
    }
  },
  {
    id: 'SUPPLIER_ID_004',
    properties: {
      supplier_name: 'Supplier Four',
```


```
      team_id: {
        data_source: { name: 'Team' },
        id: 'TEAM_ID_004',
        keyed_name: 'Logistics Team D'
      },
      effective_date: '2025-06-07T00:00:00',
      contact_number: '+1-555-400-4000'
    }
  },
  {
    id: 'SUPPLIER_ID_005',
    properties: {
      supplier_name: 'Supplier Five',
      team_id: {
        data_source: { name: 'Team' },
        id: 'TEAM_ID_005',
        keyed_name: 'Maintenance Team E'
      },
      effective_date: '2025-06-09T00:00:00',
      contact_number: '+1-555-500-5000'
    }
  },
  {
    id: 'SUPPLIER_ID_006',
    properties: {
      supplier_name: 'Supplier Six',
      team_id: {
        data_source: { name: 'Team' },
        id: 'TEAM_ID_006',
        keyed_name: 'Procurement Team F'
      },
      effective_date: '2025-06-11T00:00:00',
      contact_number: '+1-555-600-6000'
    }
  }
];
// Add sample records to the table
dataArray.forEach((data) => {
  appStudioWizard.addControlItem('dynamicTable', data.id, data);
});
```

ArasProApp Designer - Administration Guide

Introduction

Purpose and Target Audience

This guide describes how to use Aras ProAppDesigner for building complex UIs and configurations. It is addressed for administrators handling configuration of complex UIs and end users of those UIs.

Feature Terms and Definitions

The following Feature terms and definitions are used in this document:

Term	Definition

Content overview

This document describes how to use the Aras ProAppDesigner. This User Guide is organized into the following sections:

SECTION 1: **Introduction**	This section outlines the purpose and scope of this Guide, as well as its intended target audience. It supplies a brief overview of each section to help users in navigating the guide.
SECTION 2: Overview	This section provides Aras ProAppDesigner overview
SECTION 3: Architecture	This section outlines the architecture of Aras ProAppDesigner.
SECTION 4: Performance	This section provides how Aras ProAppDesigner provides better performance
SECTION 5: TOC	Aras ProAppDesigner TOC
SECTION 6: **Template Studio**	This section provides how template can be designed using studio.
SECTION 7: **Wizard**	This section provides detailed information about the wizard.
SECTION 8: **Dashboards**	This section provides information about different dashboards.
SECTION 9: **Template Customization**	This section provides information about Aras ProAppDesigner customization.

Aras ProAppDesigner Overview

Introduction

Built on Aras Innovator as an add-on solution. It is a low code solution to generate complex, responsive, mobile friendly UI. It has a rich set of controls for building complex and modern UI. Any multi-page application can be built rapidly in studio environment with drag-drop capabilities.

Organizes complex UI in to structured containers like Pages, Dialogs, Sections, Groups which intern enable UI to adapt any screen size. It enables creating guided UI for showing long data entry forms with conditional display on UI elements based on the configuration. It has rich features like conditional display, validation rules, default values, conditional formatting all with configuration. You don’t have to be a programmer to build application with Aras ProAppDesigner.

It was built using Angular SPA framework with web services as backend. It works both with Aras standard on-premise and enterprise versions.

Aras ProAppDesigner also comes with a couple of backend components, to perform certain server-side functions. These components are explained in detail in later sections of the document.

Roles

When you import Aras ProAppDesigner package, by default it deploys Template Admin group identity. This identity needs to be assigned to all users who are going to work with the templates in Template Studio.

Typical Workflow

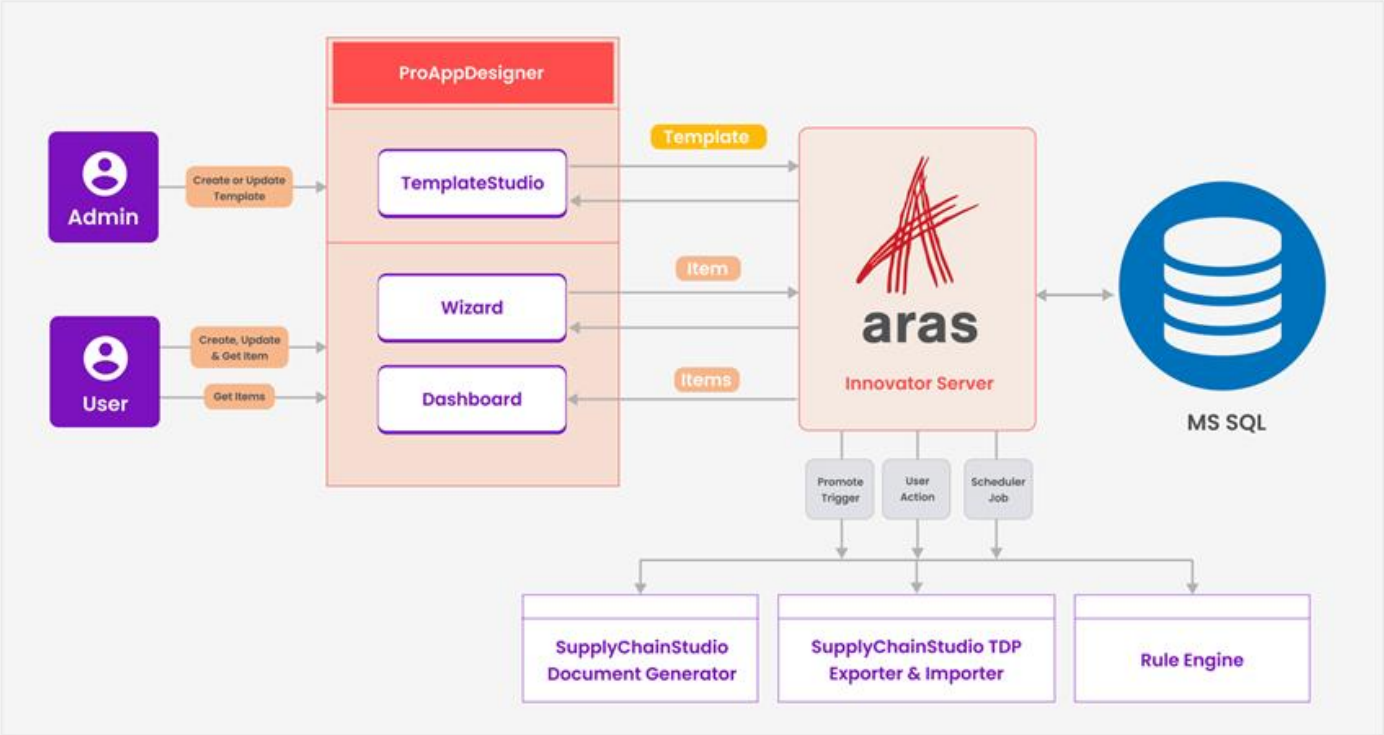
To create any application using Aras ProAppDesigner you need to follow these steps:

- Create DataModel (ItemTypes & RelationshipTypes) for the application.
- Create UI Template for the root ItemType and bind it to the DataModel.
- Publish the template through Lifecycle.
- Create, view and edit actions from Innovator.

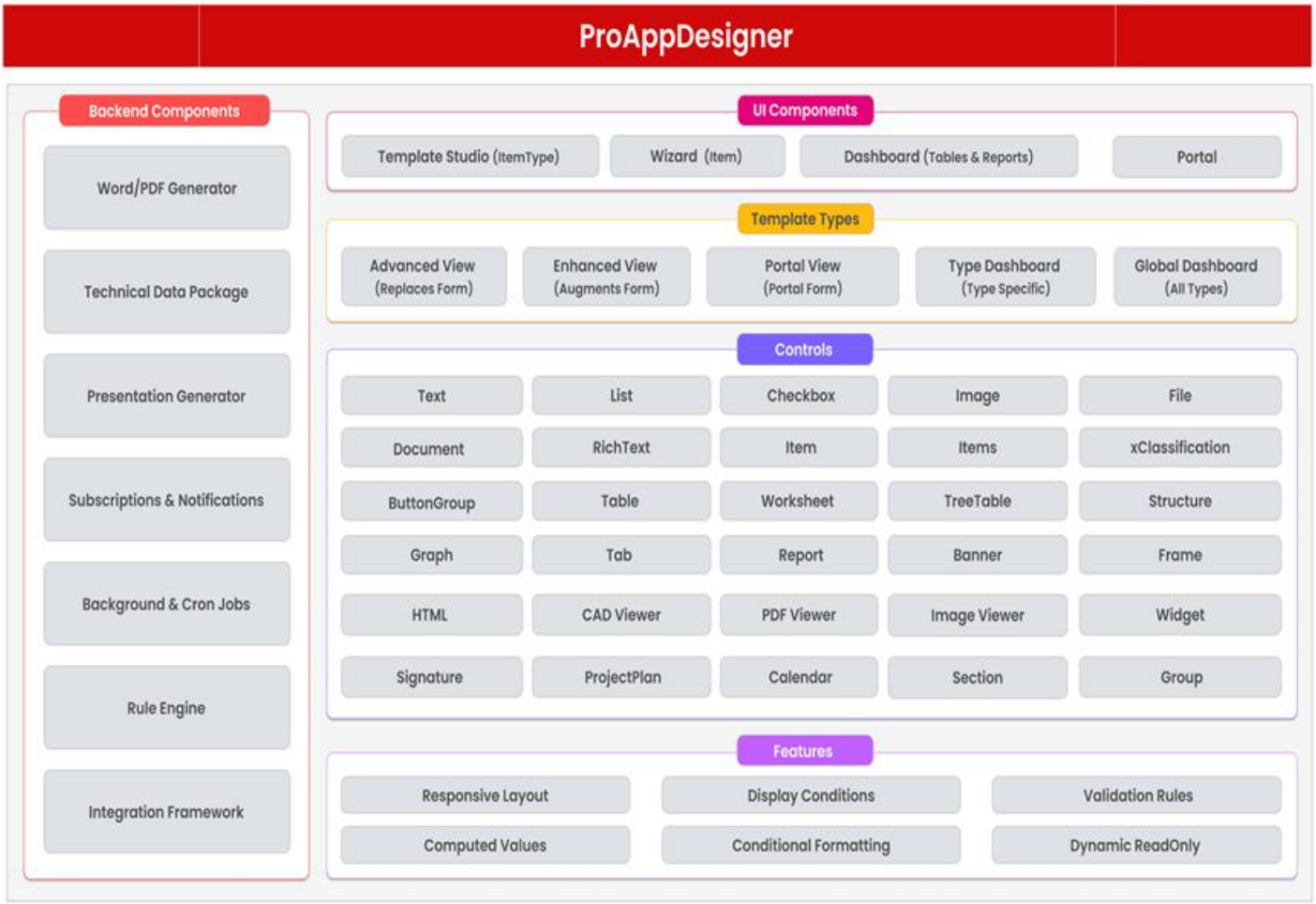
Architecture

Aras ProAppDesigner comes with two bundled SPAs, one for TemplateStudio and the other for Wizard. TemplateStudio is for creating UI template for a specific ItemType. Wizard provides a runtime view of the item that is being created or modified. Both TemplateStudio and Wizard communicate with InnovatorServer by calling ServerMethods for various actions. All the UI rendered on the client side is generated on the fly based on the data fetched from the server. Communication between client UI components and server is always lightweight (only JSON) and less chatty.

Aras ProAppDesigner Architecture:



Aras ProAppDesigner Components:



Performance

Aras ProAppDesigner code is optimized to get best possible performance by using various techniques. Since Aras ProAppDesigner is written using Angular, it will never get html from server. It generates html on the client based on data received from the server. It also uses very few roundtrips to server to get the data. Once Angular app is loaded on the client side, it will be cached on the client across browser sessions. It will be updated only when a newer version of app is available on the server.

Caching

Aras ProAppDesigner gives very good performance while rendering Studio, Wizard, and Dashboards because of the caching implemented at various levels. For showing any item using Aras ProAppDesigner, it needs Template, ItemType definition, and Item data. Template is cached in Local Storage on the client, it can persist across browser sessions. It has logic implemented to update the cache if the template is modified on the server. ItemType definition is cached at Session level, it also has logic implemented to update the cache if ItemType definition has changed on server side.

Item data is cached at session level as long as is being used in the Wizard for the showing UI. In case, same item is being shown in two different Wizards, for example one in the table other one in the form, it will have only one entry in the cache. If the item is being updated in one place, updated values will be shown in other places where it is being referred. As soon as all the references for the item in the cache are cleared from the UI, item will be removed from the cache.

Progressive Loading

While loading any Item in Aras ProAppDesigner, it gets the data for only for those controls which are visible. If any control is hidden because of display condition, its data won’t be fetched until it is displayed.

Table and Worksheet Benchmarking

This benchmarking was done on a QA machine with limited capacity. You will see better performance on a production server, since majority of the load time is taken by the server in fetching the data from the database.

Table Control

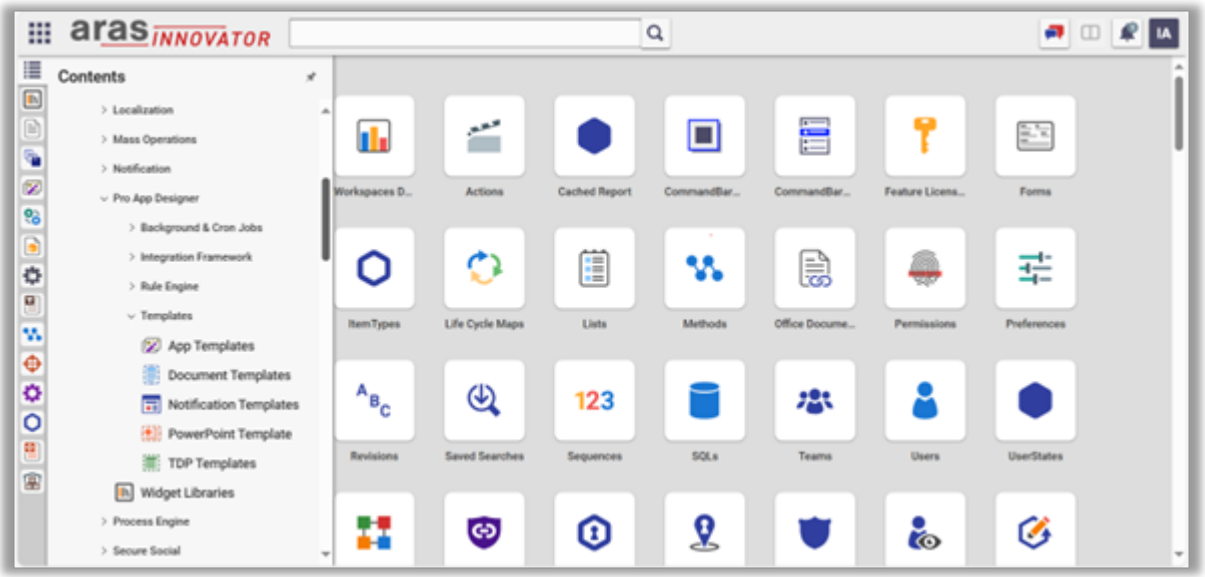
Item Count	Column Count	Pagination	Load Time
2000	8	Yes	Around 1 second
2000	8	No	Around 5 seconds
5000	8	Yes	Around 1 second
5000	8	No	Around 10 seconds

Worksheet Control

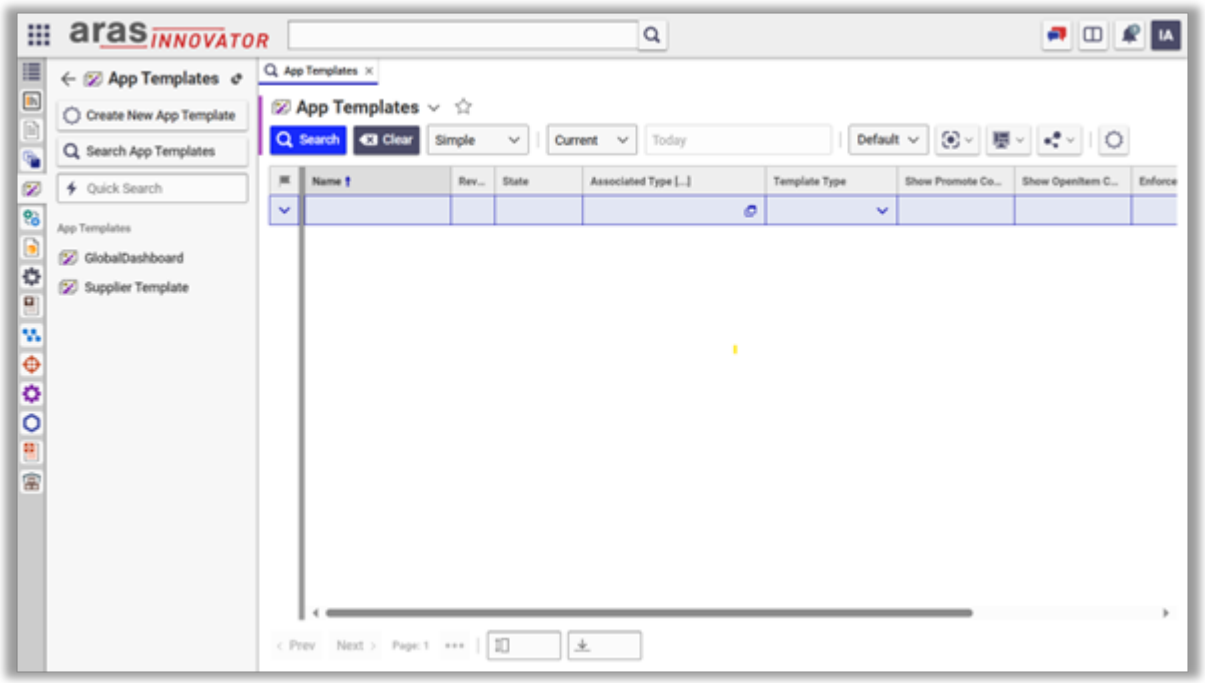
Item Count	Column Count	Load Time
2000	8	Around 4 second
5000	8	Around 8 seconds

TOC

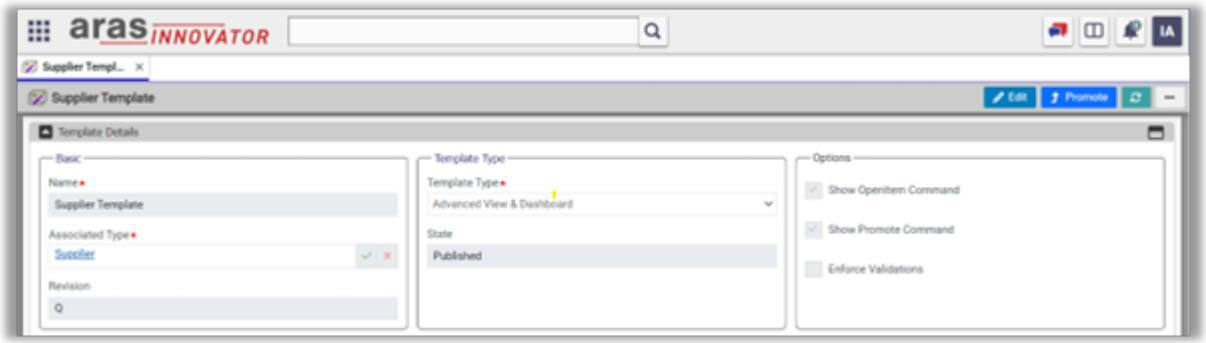
Once Aras ProAppDesigner package is installed, you will get to see a category called “Aras ProAppDesigner” under TOC. This category shows underneath AppTemplate type, upon clicking that node you will get to see all the templates existing in the system.



You can also create new template from the search grid:



For creating new template, you need provide name and select ItemType for which you want to create UI:

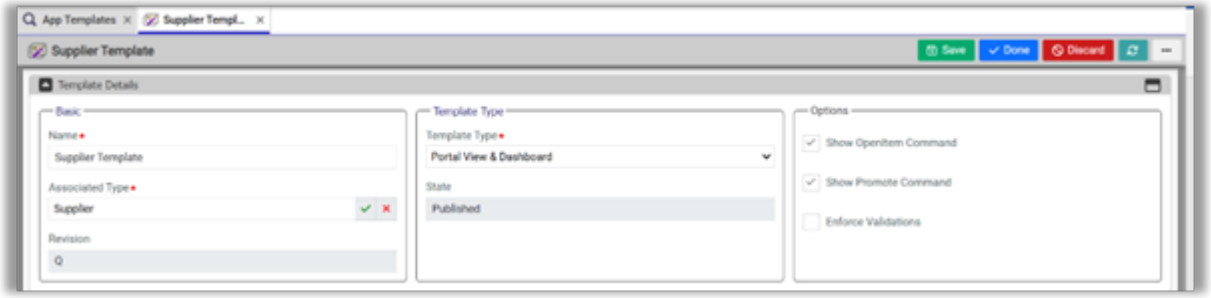


For creating template, you also need to provide TemplateType. You can select one of the following options from the dropdown based on the type of the template you want to create:

Template Type	Template Views	Description
Advanced View	Wizard, Portal Wizard	Advanced View template created for an ItemType will replace default Innovator Form.

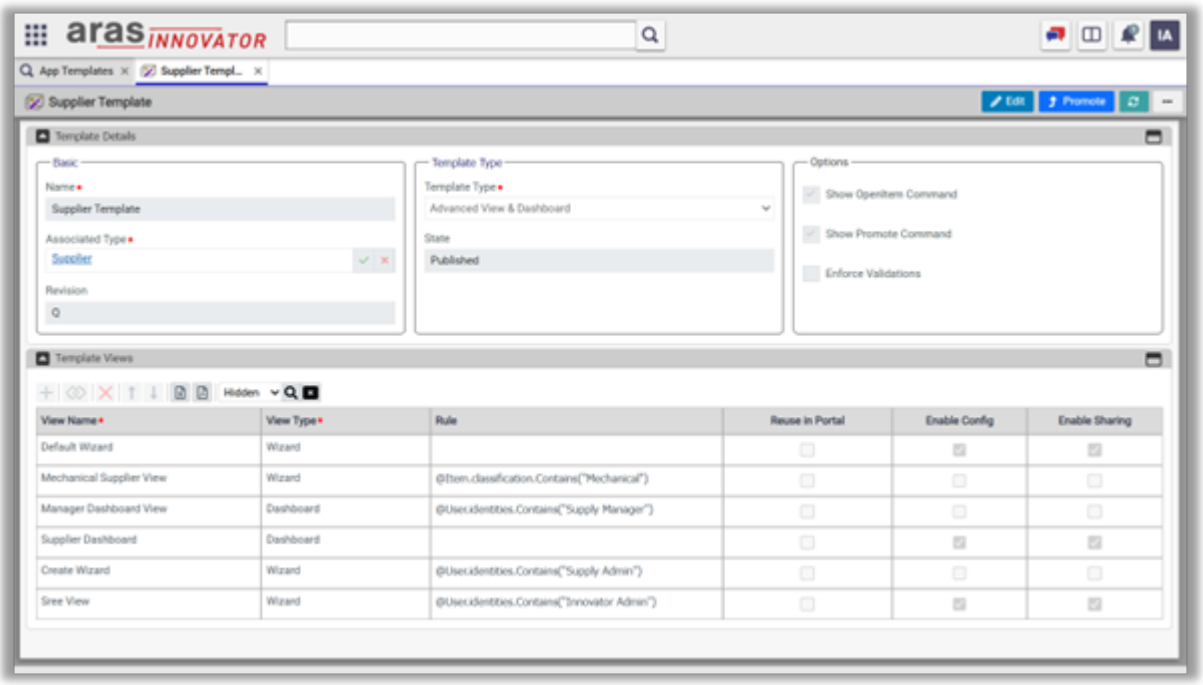
Enhanced View	Wizard, Portal Wizard	Enhanced View template will augment existing Innovator Form of an ItemType by adding Sidebar Item.
Advanced View & Dashboard	Wizard, Dashboard, Portal Wizard, Portal Dashboard	Advanced View & Dashboard template created for an ItemType will replace default Innovator Form and adds new TOC item for Type Dashboard.
Enhanced View & Dashboard	Wizard, Dashboard, Portal Wizard, Portal Dashboard	Enhanced View & Dashboard template will augment existing Innovator Form of an ItemType by adding Sidebar Item and adds new TOC item for Type Dashboard.
Type Dashboard	Dashboard, Portal Dashboard	Type Dashboard adds new TOC item for Type Dashboard that shows dashboard tables and reports of a specific ItemType.
Global Dashboard	Dashboard	Global Dashboard adds new TOC item for Global Dashboard that shows dashboard tables and reports of various ItemTypes.
Portal View	Portal Wizard	Portal View template created for an ItemType will be used to show details of the item in the ArasPortal.
Portal View & Dashboard	Portal Wizard, Portal Dashboard	Portal View & Dashboard template created for an ItemType will be used to show details of the item adds new TOC item for Type Dashboard in the ArasPortal.
Global Theme	Theme	Global Theme template allows to create styles that can be used across all templates.

Template Type Portal View & Dashboard allows to create portal view for the ItemType, that can show limited information of the items to the external users. This view is used by Aras ProAppDesigner Portal application.



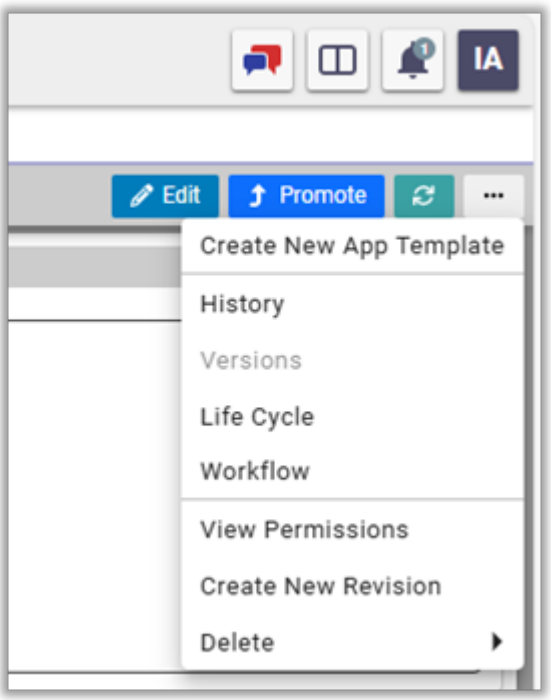
By selecting the options ‘Show OpenItem Command’ and ‘Show Promote Command’, you can control whether to show these actions in the Wizard at runtime. By selecting the option ‘Enforce Validations’, you can enforce, you can’t navigate away from the page until all validations are fixed in the current page.

After successfully creating the template, you will be able to create views for the template using “Template Views” section. By adding a row to the table and providing “View Name”, a new view will be created. For each view, you will be able to select “View Type” from the available options from the dropdown. Options will vary based on the selected “Template Type”. Please refer to the above Template Type table for the allowed View Types for each Template Type. You can provide optionally “Rule” that will define the condition when this view is used when you open or edit item from the Search Grid. Rule can be defined with a Boolean Expression which contains context ItemType properties, User properties and User roles. At runtime when you use the item, these rules across multiple views will be evaluated and first matched view will be returned to show item data. In order to allow you to create, there must be at least one view with no Rule or the Rule that doesn’t contain any ItemType properties in the expression. By selecting “Reuse in Portal” option for a view will allow reuse of the view in the Portal. By selecting “Enable Config” option, admin can allow end user to config his view, based on the default view shown to him at runtime. By selecting “Enable Sharing” option, admin can allow end user to share his view with other users in the system at runtime.



Double-clicking row or selecting it and clicking the “Open Relationship Item” command will open the view in the studio. By selecting Promote action, you will publish the template making it available for other users to use when an item is created or opened.

There are additional actions that can be performed on the template item. For example, if the template is already published, in order to update the template a new revision has to be created. You can select ‘Create New Revision’ action to create a new revision for the current template item.



Template Studio

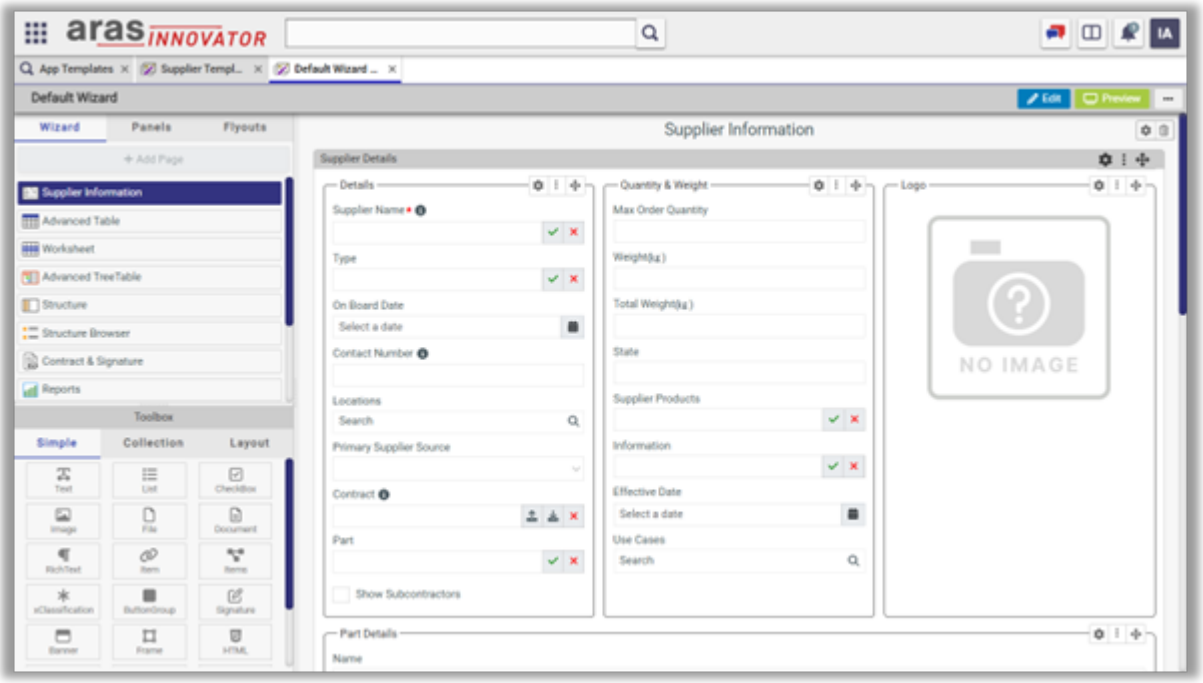
Template studio enables creating template for a specific ItemType. ItemType properties and relationships can be used to create UI of the template. Upon publishing the template, ItemType default form will be overridden with the template.

Features

- Pages and Dialogs for building single-page and multi-page applications.
- Layout controls (Sections and Groups) in Toolbox to organize controls on the pages and dialogs.
- Basic controls in Toolbox for presenting all property types from ItemType definition.
- Table control to show Related Item data in tabular form with drag & drop, conditional formatting, computed columns capabilities.
- Preview feature for reviewing changes quickly while building the template without creating Item from ItemType.
- Display Condition on all UI controls can be defined with expression to dynamically render or hide controls based on property values.
- Default Value on Text control to show computed value based on the expression built from other properties.
- Validation Rules on all controls to validate entered data before submitted request to server. Rules can be built based on the expression from the properties of the current and previous pages.
- Conditional Formatting can be defined on Text control, List control and Table control column to highlight the fields or cells with defined styles. Rules are built with expression based on thresholds set on property value.
- ButtonGroup control to define actions that can show dialogs or standard actions from Innovator like WhereUsed, StructureBrowser, DPN View etc.
- RichText control will allow you to create rich-text that can be stored as formatted text property on the item.
- Signature control can capture your signature as picture and store it as image property on the item.
- Report control can show table or chart-based reports based on QueryDefintion for local and global data.
- Structure control can show any structured data with all relationships with the help of defined QueryDefintion.
- Item control can show item property from the item. Items control can show related items from a RelationshipType with keyed names.
- Navigation panel is shown based on the defined pages and their order, provides guided UI with next and previous buttons.
- Responsive UI: Controls rendered on the page will get adjusted to the available page width to avoid showing horizontal scrollbars.

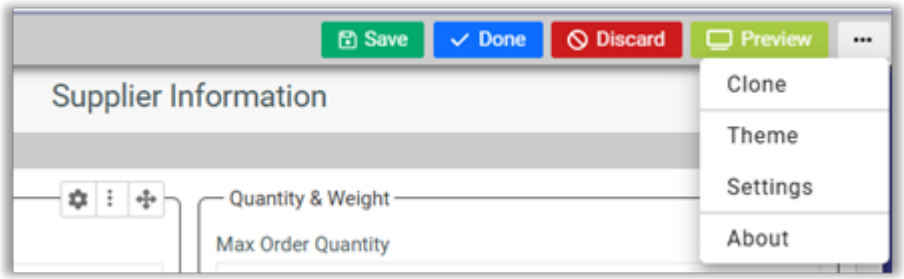
Studio Layout

When you create a new template or open an existing one, it will be shown with the following layout in the Studio. You will see at the top banner, with template name on the left and template actions on the right. On the left side, you will see sidebar with Wizard and Dialogs, where you can create and organize pages and dialogs. Beside the left sidebar, you will see Toolbox with vertically stacked controls. On the right side in the workspace, you will see selected page or dialog content.



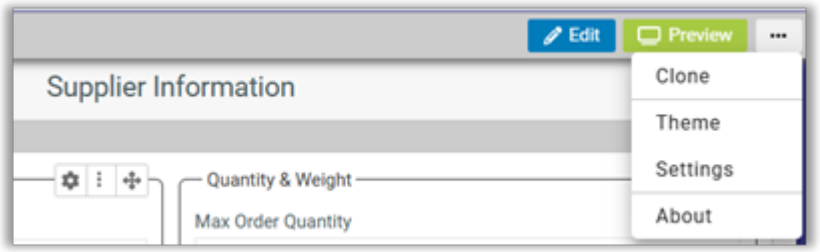
TemplateView Actions

TemplateView actions are shown in the studio in the top-right corner. When you open the template in studio with edit mode, you will see actions like Save, Done, Discard, Preview, Theme, and Help.



- **Save**: By selecting Save action, you can save the changes done so far on the template. Once changes are saved, you can't discard them using Discard action.
- **Done**: By selecting Done action, you can save the changes done so far on the template and put the template in read-only mode.
- **Discard**: By selecting Discard action, you can discard all the changes done so far on the template after the last save and put the template in read-only mode.
- **Preview**: By selecting Preview action, you can test your template in preview quickly without creating real item.
- **Clone**: By selecting Clone action, you can create a new template view based on the current view. For cloning, you need to provide "View Name" and "View Type" in the flyout.
- **Theme**: By selecting Theme action, you can set various control type styles at the template level.
- **Settings**: By selecting Settings action, you get to see template level settings in the flyout.
- **About**: By selecting About action, you get to see current Aras ProAppDesigner version and the build date.

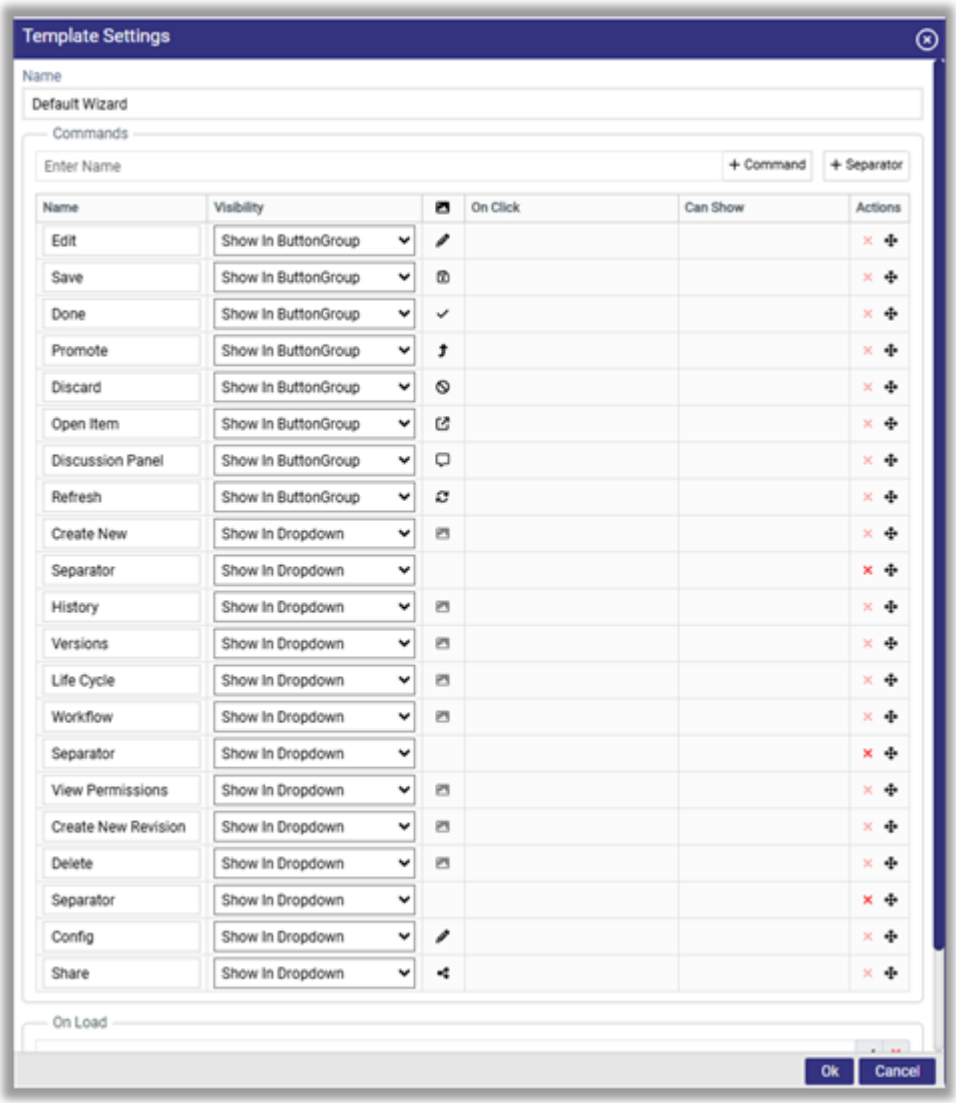
When you open template in studio with read-only mode, the following actions will be shown.



- **Edit**: By selecting Edit action, you can apply the lock on the template and get exclusive access to make changes on the template.

TemplateView Settings

TemplateView settings are shown in the flyout when you click on Settings action. Flyout shows TemplateView name, commands that can be shown at runtime on top of item details page. It also allows you to configure events like OnLoad and OnSave.



- **Name**: It shows the name of the current TemplateView. It is shown as read-only.
- **Commands**: It allows you to configure actions shown on the top of item details page. By default, it shows standard commands for which you can change visibility and icons. With visibility setting, you can configure the action to be shown as part of ButtonGroup or Dropdown. You can hide the action by selecting visibility as Hide. You can also add custom commands by defining “On Click” and “Can Show” scripts.

- **On Load**: OnLoad event allows you to write custom logic that will be executed after the item date is loaded. Please refer to the Template Customization section for more details.
- **On Save**: OnSave event allows you to write custom logic that will be executed after the item is saved. Please refer to the Template Customization section for more details.

Control Actions

Every control has following action icons shown upon selecting the control. These actions are available on the control only when the template is in the edit mode:

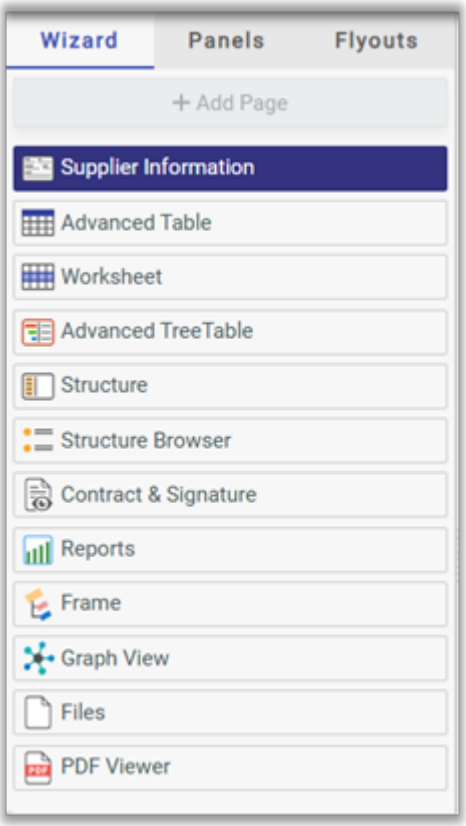
- **Settings**: By clicking the settings icon, settings flyout will open on the right side from where you can change various settings of the selected control.
- **Miscellaneous**: By clicking three-dot icon, two actions will show in the dropdown. Using delete action, you can delete the selected control from the page. Using MoveTo action, you can move the control in the current page or to any other pages under different parent.
- **DragDrop**: By holding drag-drop icon, you can reposition the selected control within its immediate parent.

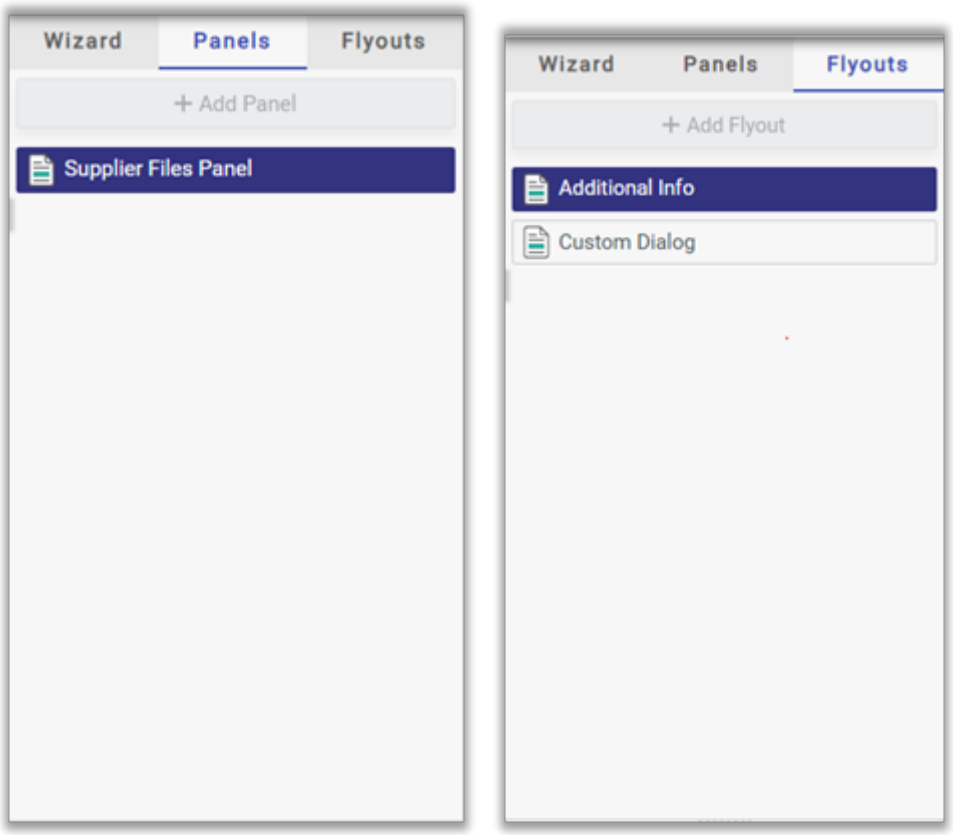
Template Studio Sidebar

Template Studio Sidebar shows Wizard, Panels & Flyouts in the upper section and Toolbox in the lower section. Wizard tab shows a list of pages that can be shown at runtime when you open the item. In Aras ProAppDesigner you can create single page or multi page applications. When you create only one page, at runtime when the page is shown in the Wizard, it shows item's keyed name as page title. In case of multipage application, all the pages are shown in the Navigation bar with their page titles.

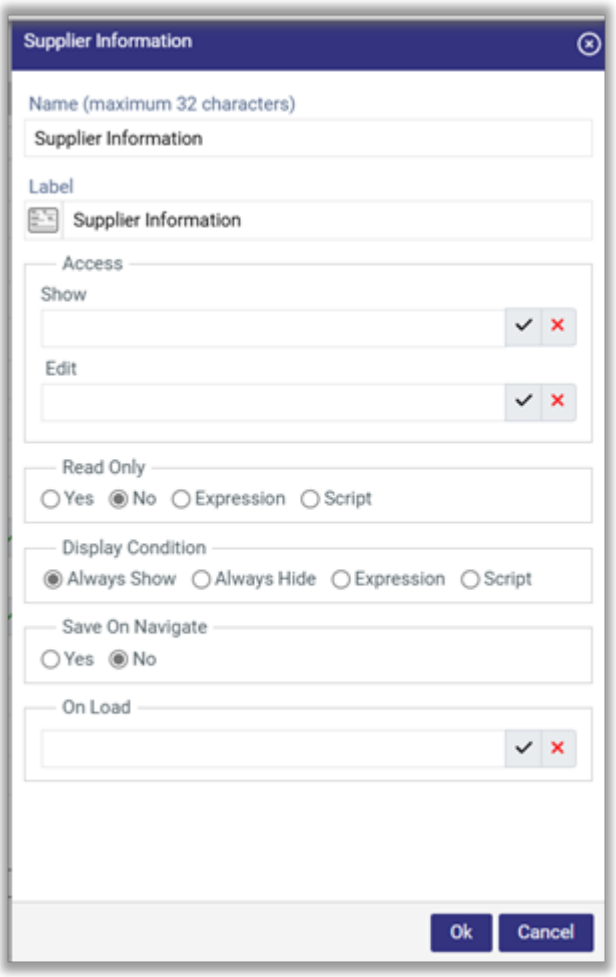
Panels tab shows list of panels that can be used while adding panel control in any group control on the page. Panel looks like a page in all aspects, and it can be created just like a page with only exception that it can be used under a group control in the page to create complex layouts. Same panel can be used on many pages, for showing same header and footer across all pages of the Wizard.

Flyouts tab shows a list of flyouts that can be used at many places in the Wizard by associating them to click handlers of various controls like buttons and toolbar commands. Flyout looks like a page in all aspects, and it can be created just like a page with only exception that it can be shown as a popup with submit, cancel and reset buttons.





You can create new page by clicking on ‘Add Page’ button in the sidebar. You can drag-drop pages to reorder created pages. Each page on the right side is shown with actions in the top-right corner with settings and delete icon.

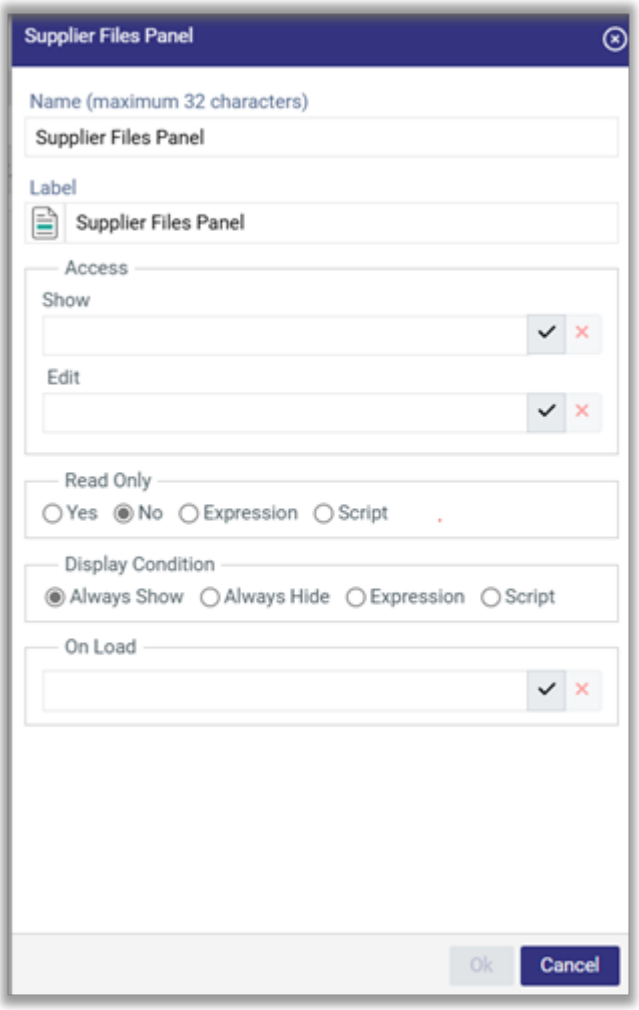


- **Name**: It is used to uniquely identify page inside the wizard.
- **Label**: It is used to show title on the page. It can be localized by defined Languages and Locales inside innovator. Label also allows you to set the icon for the page.
- **Access**: It allows you to define roles which can access or edit the page. Roles (group or alias identities) can be at Show and Edit level. When page is hidden because you do not have defined role, properties used inside this page are not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, whole page can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the page read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If the returned value is true, the whole page will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the page will be shown or hidden from the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the page if expression is evaluated to true. Expression will be evaluated based on the cached property

values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, whole page will be hidden.

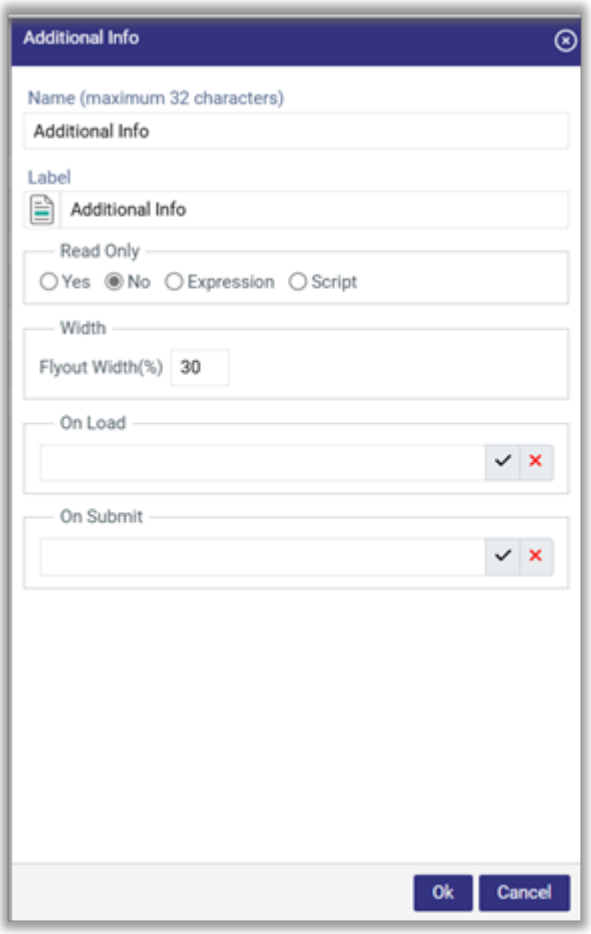
- **Save On Navigate**: If this option is selected as Yes, all updates made so far are saved to server when you navigate away from the page.
- **OnLoad**: OnLoad event allows you to show custom UI in the page if the page has HTML or Frame controls. Please refer to the Template Customization section for more details.

Panels can be created or managed in the same way as wizard pages. Panel can be placed just like any other control under the group through panel control available in the Toolbox. Panel can also be used as a tab inside Tab control.



Flyouts can be created or managed in the same way as wizard pages. The difference is, flyout can be shown as a popup at runtime based on the button click event. In order to show flyout as a popup,

you need to place a button on the page and associate its OnClick handler with the flyout. Flyout has one additional setting to set the Flyout Width in percentage with respect to the page width.

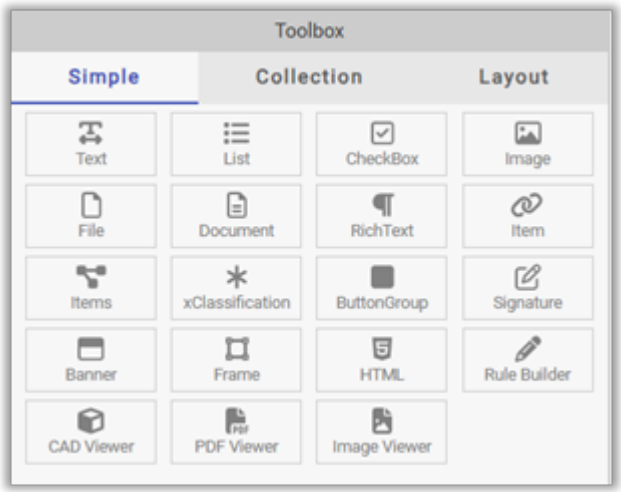


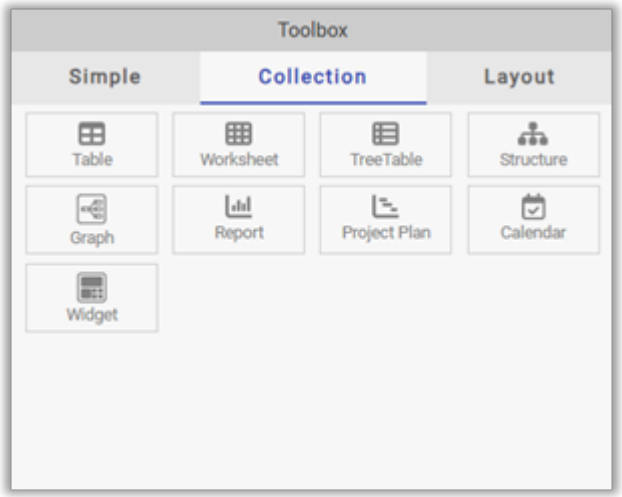
- **OnSubmit**: OnSubmit event allows you to write custom logic that will be executed when the page is submitted. It can be mainly used if the dialog has any custom UI and if you want to save the changes to page or any external system. Please refer to the Template Customization section for more details.

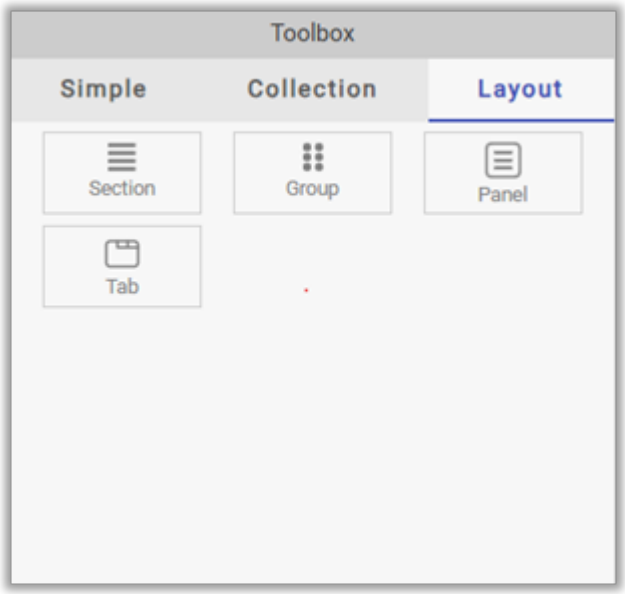
At runtime, flyout is shown as popup, it shows Reset, Ok and Cancel buttons at the bottom. Reset button will reset all values to their original values if you made any changes, Ok button will commit the changes to the client cache and Cancel button will close the flyout without committing the change to the cache.

Toolbox

Toolbox is shown vertically in the studio with all available controls. In order to place any control in the page, you need to select container where you need to place the control. When container or control is selected from the page, only those controls which can be added to the page with respect to selected control will be enabled in the Toolbox. Each control from the Toolbox will be explained in detail later in the following sections.

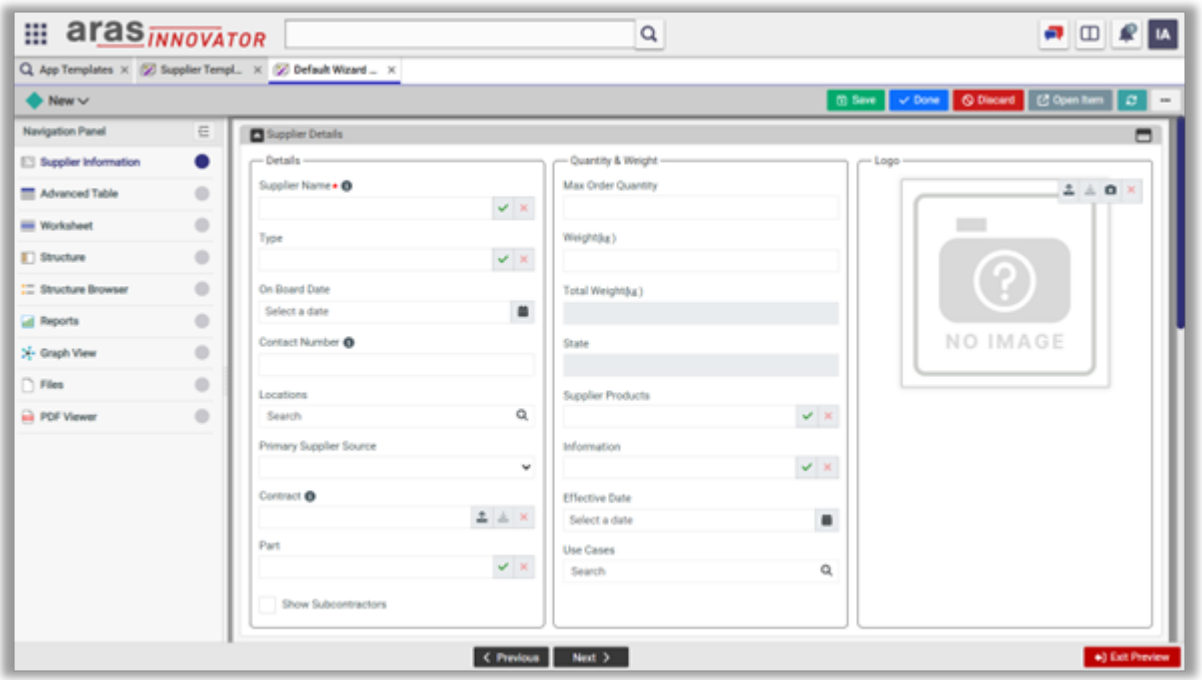






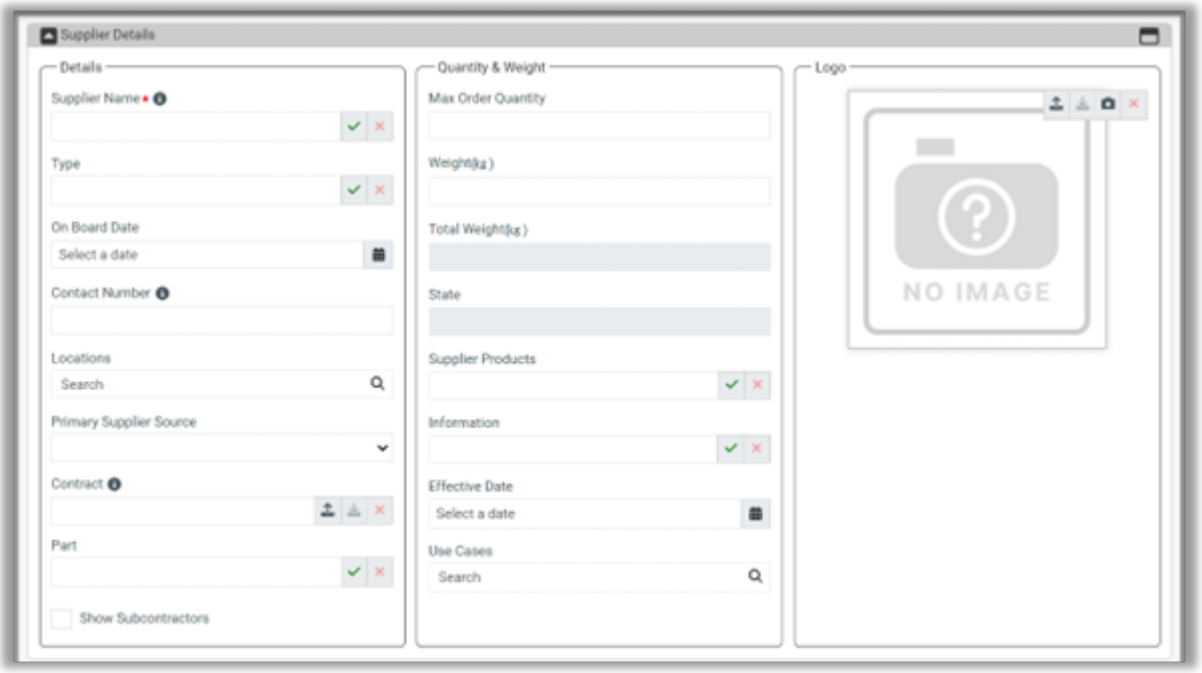
Template Preview

When you select Preview action, it shows how wizard looks at runtime when you really work with the item without creating the real item. It is very handy to test all your changes quickly without creating the item before publishing the template. In preview mode, you can check how your pages are shown with their UI with all display conditions, validation rules, default value (computed fields), and conditional formatting. In case any specific control can’t be rendered because of missing context item, it shows the message indicating the same.

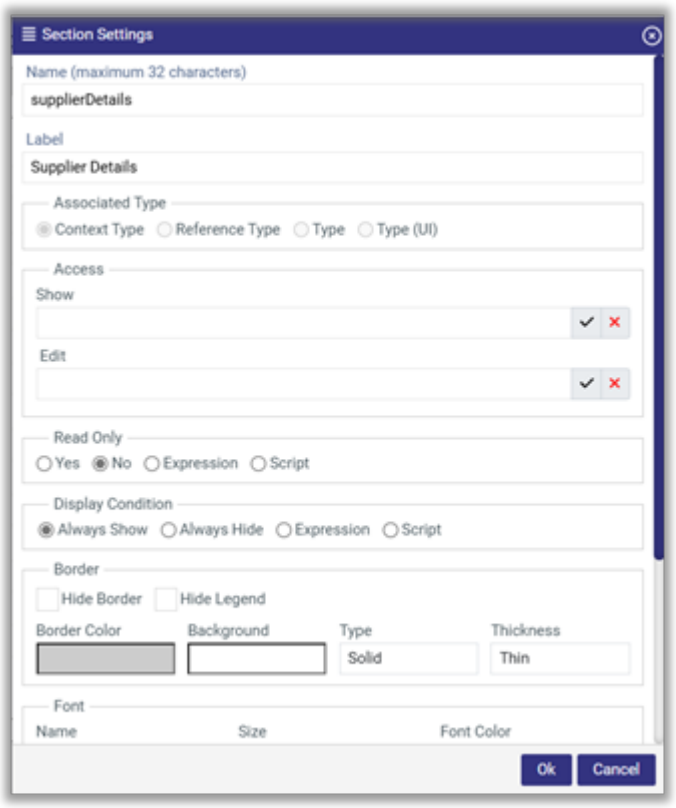


Section Control

Section control is a layout control to organize group controls placed inside it. Section controls are vertically stacked one over the other. Section controls are the top-level container controls that exist in pages and dialogs. By defining display condition or access permissions, section control can be shown or hidden based on the evaluation criteria. When section control is created on the page, by default a group control is placed inside it.



Section control behavior or characteristics can be configurated through its settings flyout by selecting gear-icon from top-right corner. You can place the section anywhere on the page among other sections using drag-drop icon. By selecting 3-dot icon, it shows Delete and MoveTo options to delete or move the section to other pages.



- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show the title of the section. It can be localized by defined Languages and Locales inside innovator.
- **Associated Type**: Following options can be selected for the Associated Type. By default, Associated Type is set as Context Type.

    - **Associated Type:** Context Type

      When Context Type is selected, while adding control to a group of sections, properties from the template ItemType will be shown for binding with the control.

    - **Associated Type:** Reference Type

      When Reference Type is selected, a dropdown with a list of all item properties from the contextual ItemType will be shown. By selecting a value for the dropdown, you will be shown with properties of the selected ItemType inside the groups of the section for binding with controls.

    - **Associated Type:** Type

When Type is selected, while adding a control to a group of sections, properties from the selected ItemType will be shown for binding with the control. In this case, the section will be shown in the UI only when the item of the selected ItemType is set as context item on the section.

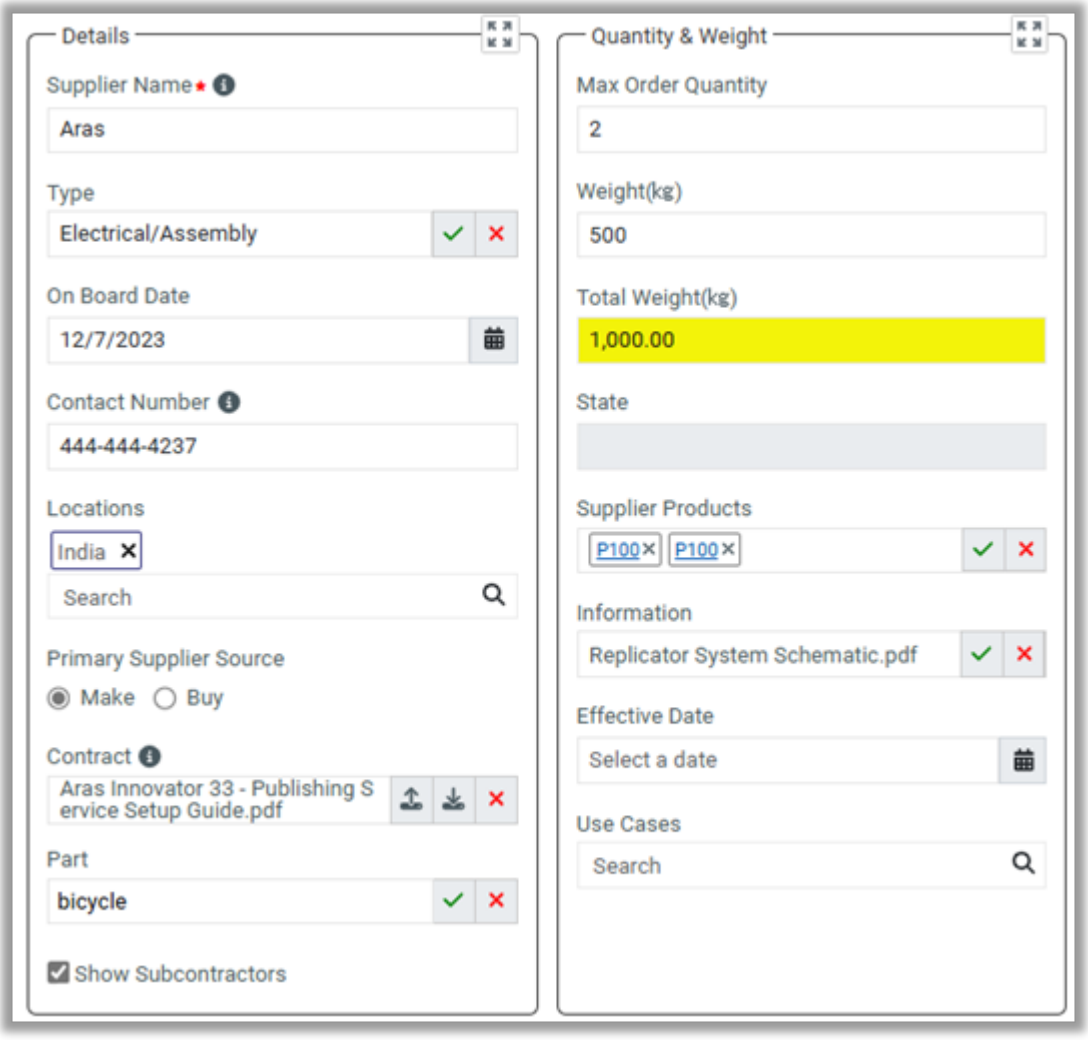
- **Associated Type:** Type (UI)

  When Type (UI) is selected, while adding control to a group of sections, properties from the selected ItemType will be shown for binding with the control. In this case, the section will always be shown in the UI with empty values. Selected ItemType will only be used to show custom form with properties defined on the selected ItemType. You must write your own custom script to save these values to the server.

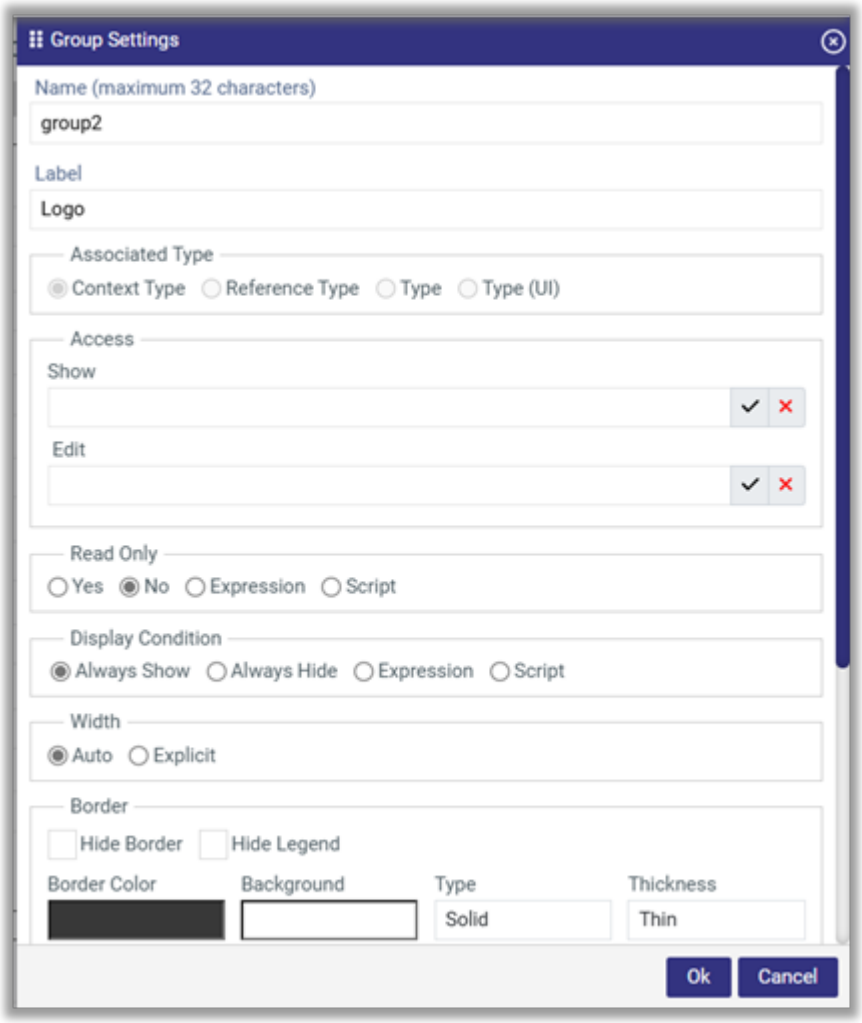
- **Access**: It allows you to define roles which can access the section. Roles (in the form group identities) can be at Show and Edit level. When section is hidden because you do not have defined role, properties used inside this section are not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, section can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the section read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If the returned value is true, the whole section will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the section will be shown or hidden from the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the section if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If the returned value is false, the whole section will be hidden.
- **Border**: It allows you to show or hide the border and legend of the section. In case if you choose to show the border, you can select the border style using Color, Background, Type and Thickness settings. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **Font**: It allows you to set the required font, size, color for showing the legend of the section. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **Margin**: It allows you to set required top, right, bottom, left margins with respect to adjacent controls. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **Padding**: It allows you to set the required top, right, bottom, left padding with respect to parent control. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.

Group Control

Group control is a layout control used to organize child controls placed inside it. Group controls are horizontally stacked beside each other inside a section control. When multiple group controls are placed inside section control, they equally take the available width. When you resize the page on desktop, based on the available width they can be shown in single row or multiple rows inside a section control. In mobile view, they will be vertically stacked inside a section control. By defining display condition or access permissions, group control can be shown or hidden based on the evaluation criteria. When a section control is created on the page, by default a group control is placed inside it.



Group control behavior or characteristics can be configurated through its settings flyout by selecting gear-icon from top-right corner. You can place the group anywhere inside the section among other groups using drag-drop icon. By selecting 3-dot icon, it shows Delete and MoveTo options to delete or move the group within current page or to other pages.



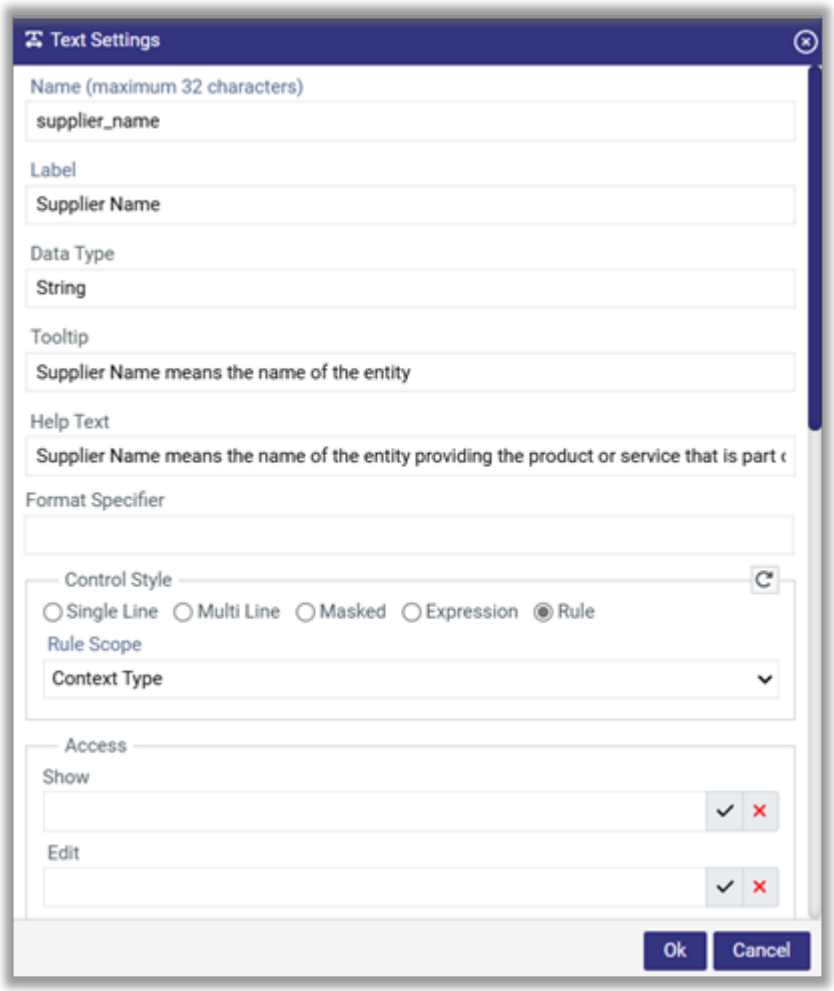
- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show the title of the group. It can be localized by defined Languages and Locales inside innovator.
- **Associated Type**: Following options can be selected for the Associated Type. By default, Associated Type is set as Context Type.
    - **Associated Type:** Context Type
      When Context Type is selected, while adding control to a group of sections, properties from the template ItemType will be shown for binding with the control.

  - **Associated Type:** Reference Type
    When Reference Type is selected, a dropdown with a list of all item properties from the contextual ItemType will be shown. By selecting a value for the dropdown, you will be shown with properties of the selected ItemType inside the groups of the section for binding with controls.
  - **Associated Type:** Type
    When Type is selected, while adding a control to a group of sections, properties from the selected ItemType will be shown for binding with the control. In this case, the section will be shown in the UI only when the item of the selected ItemType is set as context item on the section.
  - **Associated Type:** Type (UI)
    When Type (UI) is selected, while a adding control to a group of sections, properties from the selected ItemType will be shown for binding with the control. In this case, the section will always be shown in the UI with empty values. Selected ItemType will only be used to show custom form with properties defined on the selected ItemType. You must write your own custom script to save these values to the server.
- **Access**: It allows you to define roles which can access the group. Roles (in the form group identities) can be at Show and Edit level. When group is hidden because you do not have defined role, properties used inside this group are not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, group can become read-only even if user can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the group read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the whole group will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the group will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the group if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, whole group will be hidden.
- **Width**: It allows you to set the width for the group. Width can set as ‘Auto’ or ‘Explicit’. If it is set as ‘Auto’, the parent section will give equal widths for all the groups defined with the ‘Auto’ option from the available width. If it is set an ‘Explicit’, it gives you the option to set the width in percentage from the available width of the parent section. All the groups defined with ‘Explicit’ option under the same section will first be given requested widths from the available width, and remaining groups with ‘Auto’ option will get equal portions from the remaining width.
- **Border**: It allows you to show or hide the border and legend of the group. In case if you choose to show the border, you can select the border style using Color, Background, Type and Thickness settings. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **Font**: It allows you to set the required font and size for showing the legend of the group. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **Margin**: It allows you to set required top, right, bottom, left margins with respect to adjacent controls. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **Padding**: It allows you to set the required top, right, bottom, left padding with respect to parent control. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **On Save**: OnSave event allows you to write custom logic that will be executed when the data shown in the group is saved. It can be mainly used if the group shows the data related to an item that is different than the item shown on the page. Please

refer to the Template Customization section for more details.

Text Control

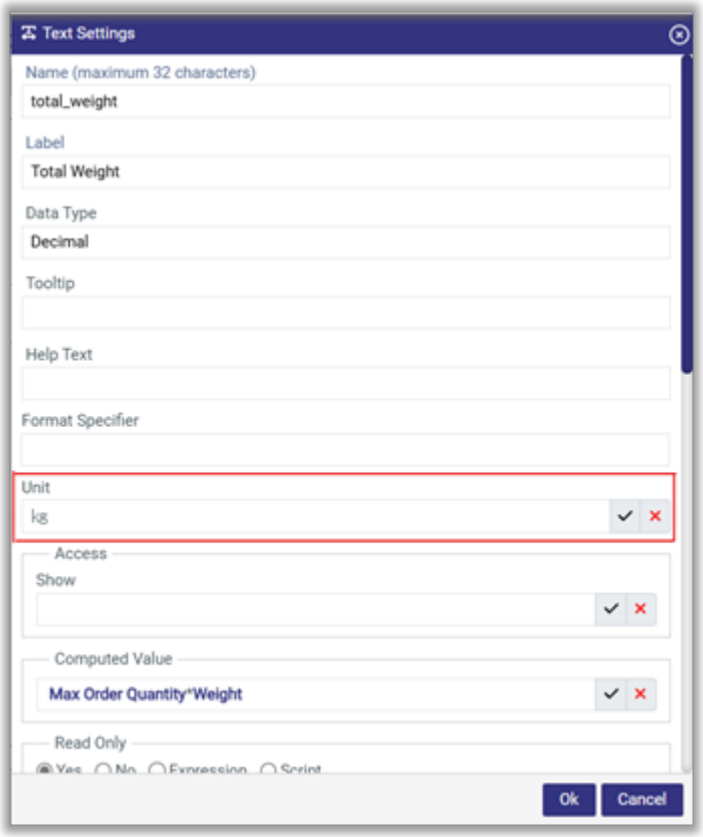
Text control displays properties of datatypes String, Text, Integer, Float, Decimal, Date, Sequence and Classification. Text control honors Format Specifier set on the ItemType definition. Text control honors the default value set through ItemType definition. You can also set default value for Text control by defining text expression with properties from other controls from the current page or previous pages. Text control has additional settings like Display Condition, Validation Rules, Conditional Formatting etc. that are covered in the subsequent sections.

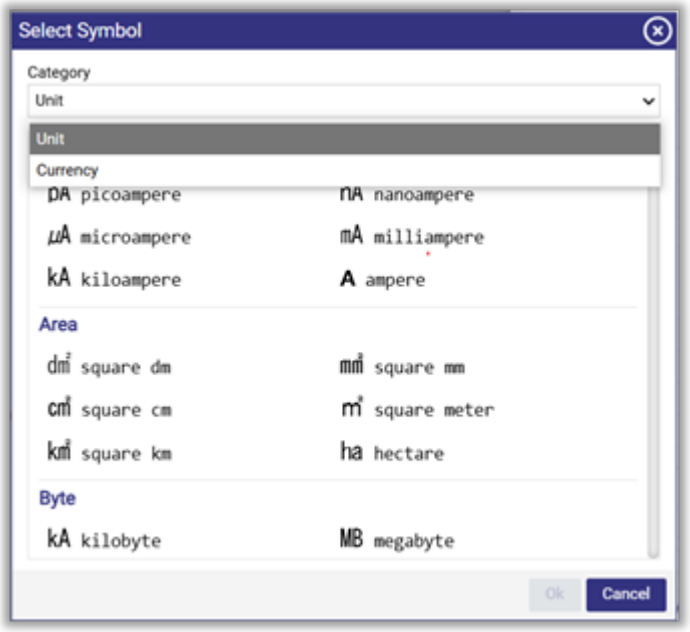


- **Name**: It is used to uniquely identify control inside the template. For text control, name is derived from the property bound to the control.
- **Label**: It is used for showing title on the text control. For text control, label is derived from the property bound to the control. If property label is localized inside innovator, at runtime localized label is pulled from the property definition.

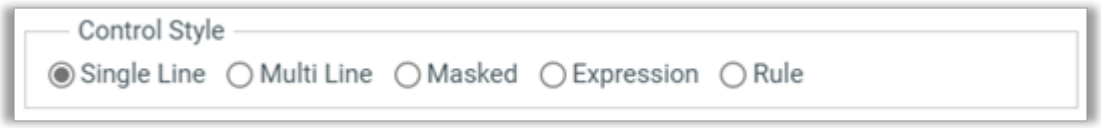
- **Data Type**: It allows type of data that you can entry in the text control. It is derived from the property bound to the control.
- **Tooltip**: Tooltip will be shown when you hover the mouse on the information icon on the form field. It is derived from the property bound to the control.
- **Help Text**: Help Text flyout will be shown when you click on the information icon on the form field label. It is derived from the property bound to the control.
- **Format Specifier**: It shows the format in which data should be rendered in the control. It is derived from the property bound to the control.

- **Unit**: Unit setting will be shown only for the numeric properties associated with the control. Using this setting, you should be able to set units for Weight, Length, Area etc. through the Select Symbol dialog.

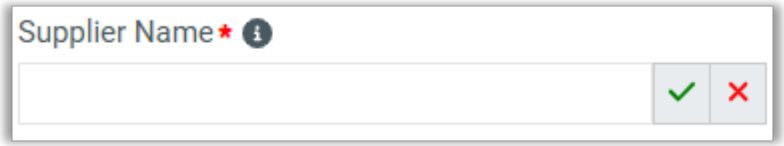




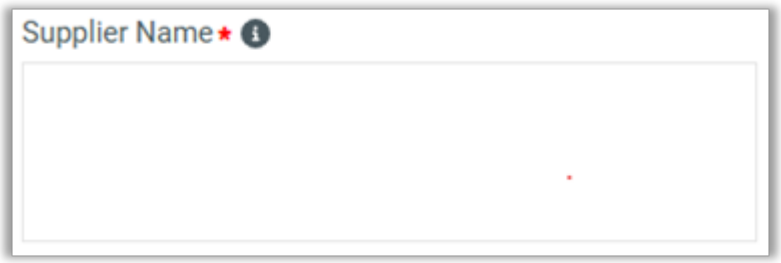
- **Control Style**: Text control can be rendered as textbox, text area, rule by setting its style as single line, multi-line, masked, expression, or rule.



Control Style: Single Line

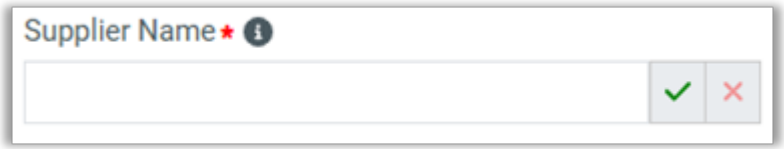


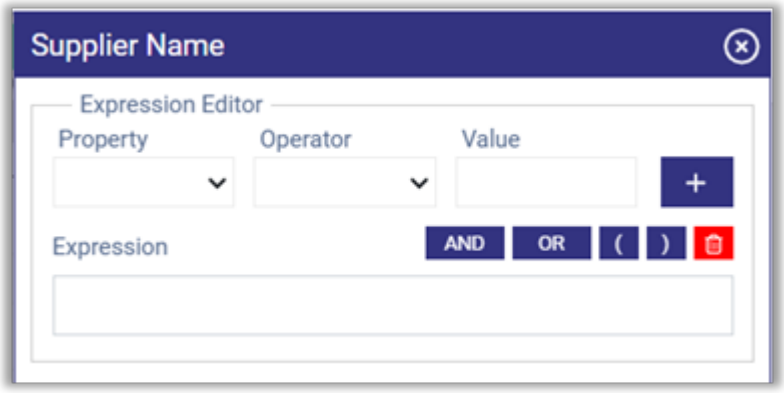
Control Style: Multi Line



- **Control Style:** Masked

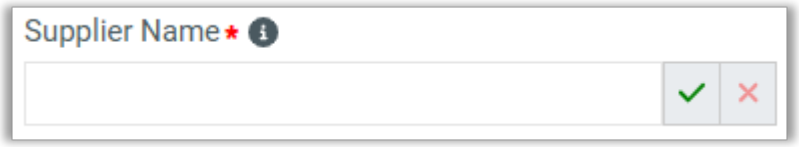
Control Style: Expression
With expression style, editing the control will show expression editor to create expression with properties from the contextual ItemType.

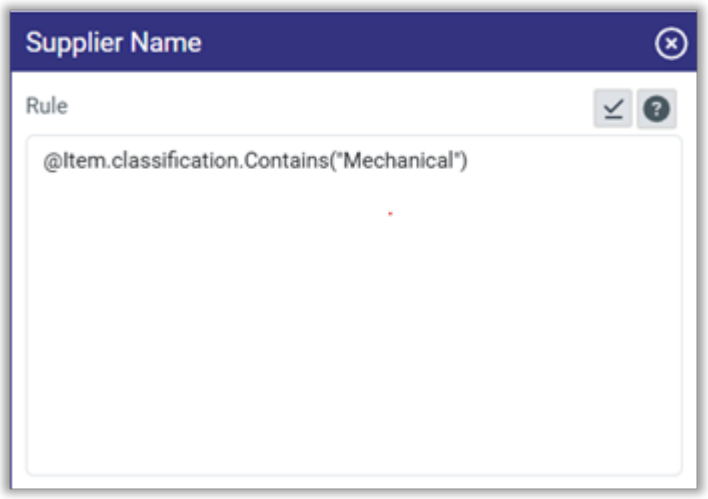




Control Style: Rule

With rule style, editing the control will show rule editor to create rule with properties from the contextual ItemType.



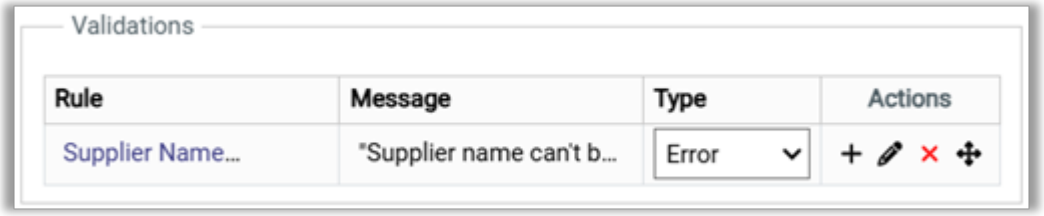


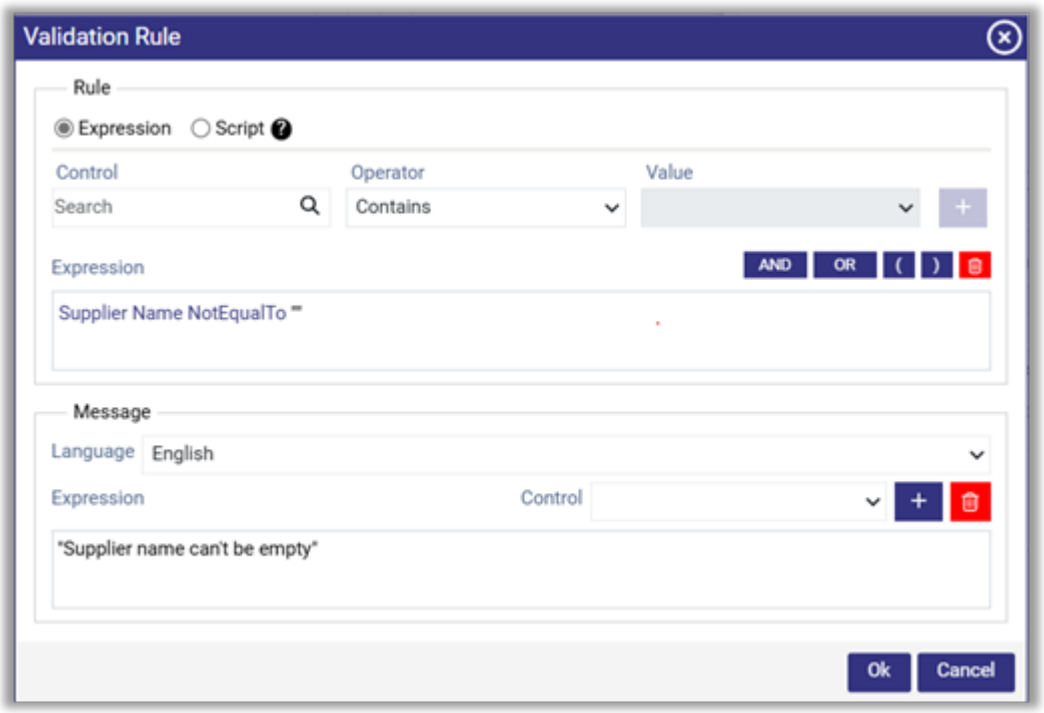
- **Access**: It allows you to define roles which can access control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, property associated with this control is not even fetched from server.
- **Computed Value**: It allows you to show default value when control is rendered. Default value can be computed based on the expression; in the expression you can reference other properties from the current page or previous pages. If you create an expression for default value and make control as read-only, it can dynamically change its value based on what you enter in other controls on the page.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If the returned value is false, the control will be hidden.

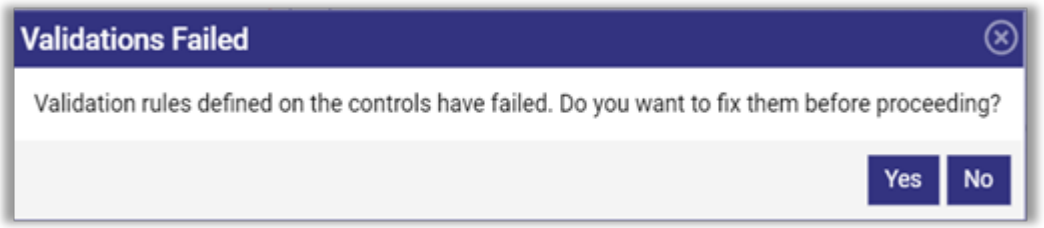
- **Validations**: By default, a row is displayed under the Validations section. By clicking the cell under Rule, a condition or script can be set to validate the data entered in the control. Normally validations are executed when you click on Save, Done, Next buttons from the wizard. While

you are defining the rule, you need to set the message that you want to display to user when rule is not satisfied. Message can also be a text expression; you can use current control or other control values to display dynamic text to the user. You can also set locale specific messages for the languages you configured. While defining the rule, you also need to specify Type of the rule as Error or Warning. Additional rules can be added by clicking on Add (+) action on the previous row after which you need to add the rule. If you define multiple rules, all rules should be satisfied to move forward. You can remove or reposition the rules using the respective icons from actions column.

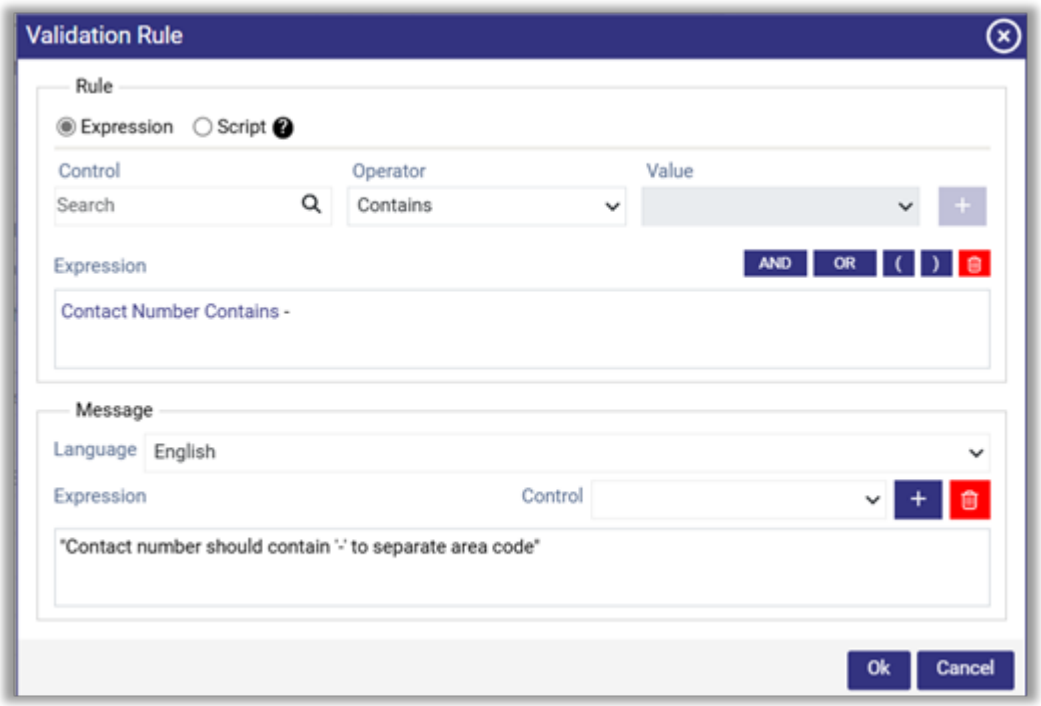
The following example shows validation rule of type Error. When validation rule fails, error will be shown in the popup dialog with options to ignore and move forward or fix the issues before moving forward. If you move forward without fixing validation errors for that page, in the navigation panel, page title with be shown with red circle with tick icon to indicate there are validation errors in the page.

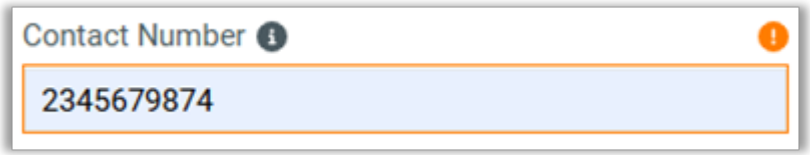


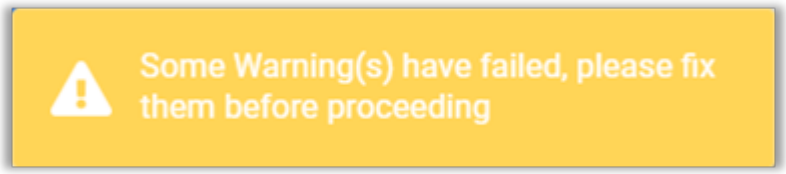




The following example shows validation rule of type Warning. When validation rule fails, warning will be shown in the flash notification with orange background color. Warnings can be ignored by clicking on Save, Done, Next buttons second time to move forward with the warning.



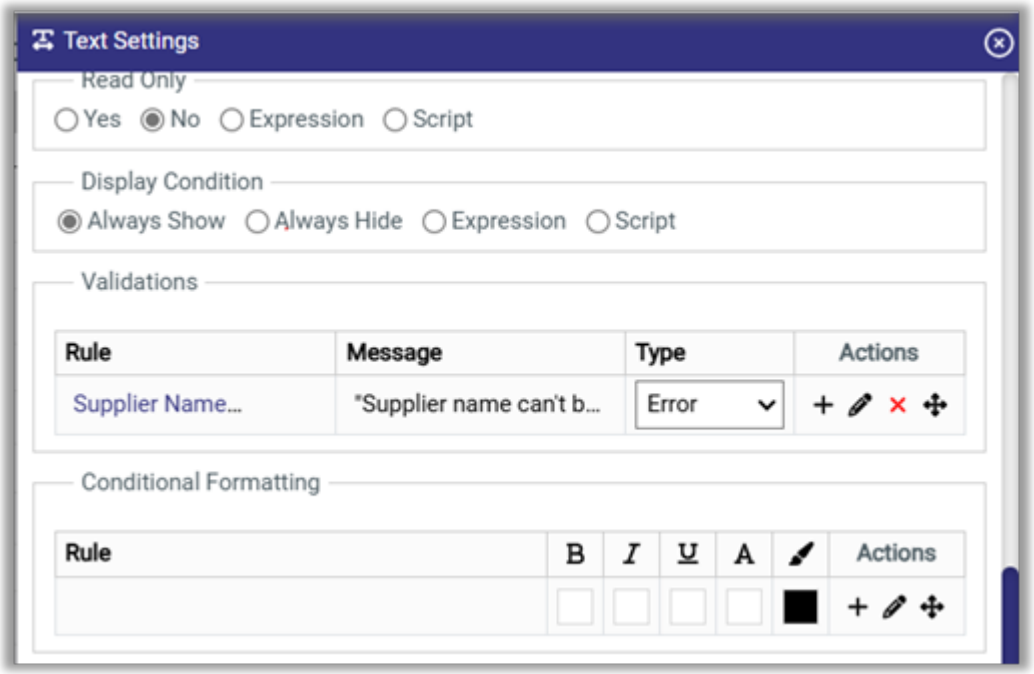


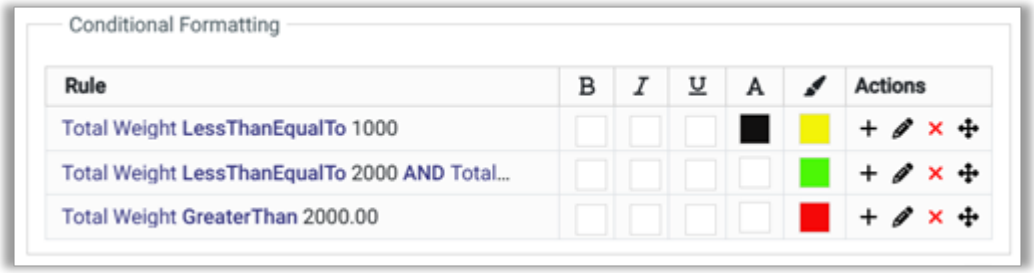


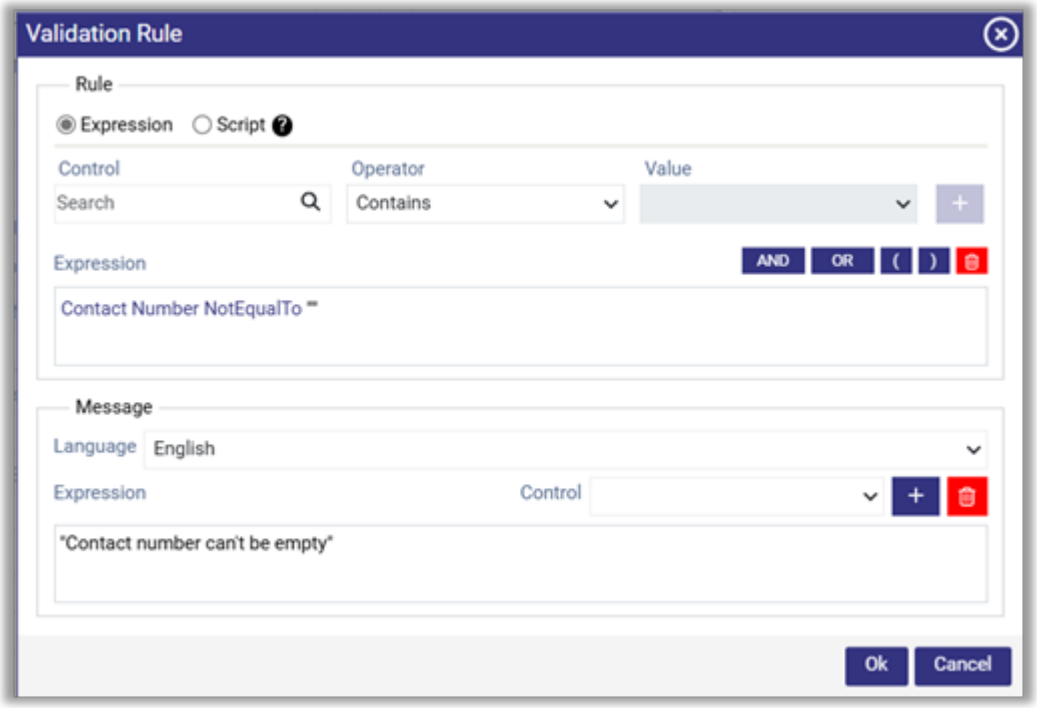
Validation rule can be defined using Script option. This will be useful for defining validation based on some complex logic rather than simple expression. Script can be written in JavaScript by accessing control values using API. Script can contain the logic to access data from the server to perform complex validation. Script should return a Boolean value to indicate whether the validation is successful.

- **Conditional Formatting**: By default, a row is displayed under Conditional Formatting section. By clicking the cell under Rule, a condition or script can be set to display specific styles for the data displayed in the control if the condition is evaluated to true. Additional rules can be added

by clicking on Add (+) action on the previous row after which you need to add the rule.
Condition formatting rules are evaluated at runtime in the order in which they are defined from first to last, whenever the first rule is satisfied it stops and won't execute remaining rules. Because of that, extra care should be taken while defining the order of the rules.
You can edit the existing rule by clicking on action icon null . You can remove or reposition the rules using the respective icons from actions column.







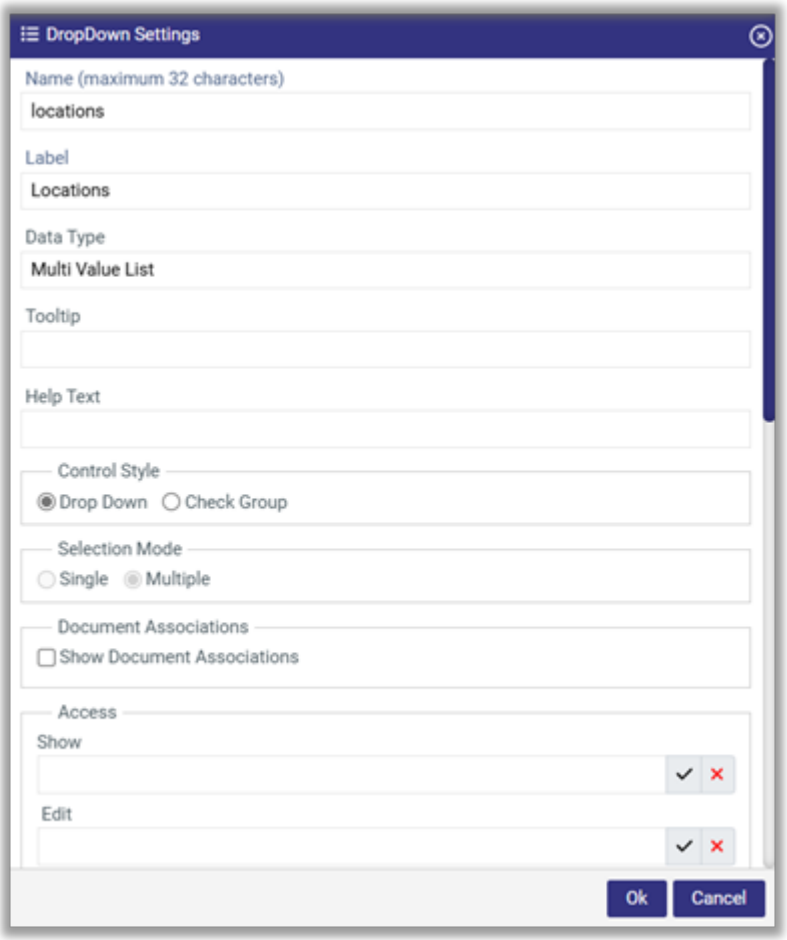
Conditional Formatting rule can be defined using Script option. This will be useful for defining the rule based on some complex logic rather than simple expression. Script can be written in JavaScript by accessing control values using API. Script can contain the logic to access data from the server to execute complex rule. Script should return a boolean value to indicate whether the rule is successful.

- **Reset Dependency**: It allows you to reset value of the control to the default if any of the controls defined in “Reset Dependency” have their value changed at runtime.
- **Font**: It allows you to set the required font and size for the value in the control. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **Margin**: It allows you to set required top, right, bottom, left margins with respect to adjacent controls. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the value in the control is loaded. Please refer to the Template Customization section for more details.

- **On Change**: OnChange event allows you to write custom logic that will be executed after the value in the control is updated. Please refer to the Template Customization section for more details.

List Control

List control can be used to display properties of datatypes List, Filter List and Multi Value List. While displaying list, Aras ProAppDesigner uses same underlining list with options defined in the Innovator. List control comes with Control Style setting, with this, options can be shown as RadioButtonGroup in case of single selection list or CheckboxGroup in case of multi selection list. If you don't set the Control Style, default Dropdown will be used to show the options. List control honors the default value set through ItemType definition.

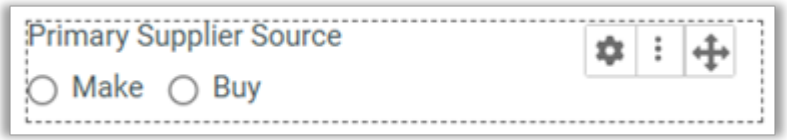


- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show label on the control. It can be localized by defined Languages and Locales inside innovator.
- **Data Type**: It shows the type of the property as defined in the ItemType definition. It can be List, Filter List or Multi Value List, and it is shown as disabled.

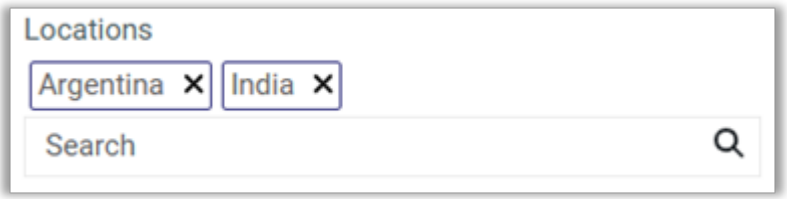
- **Tooltip**: Tooltip will be shown when you hover the mouse on the information icon on the form field. It is derived from the property bound to the control.
- **Help Text**: Help Text flyout will be shown when you click on the information icon on the form field label. It is derived from the property bound to the control.
- **Control Style**: You can configure List control style as Radio Group in case of List or Check Group in case of Multi Value List.

When Radio Group or Check Group is selected as a style, Orientation setting will be shown to arrange options either horizontally or vertically.

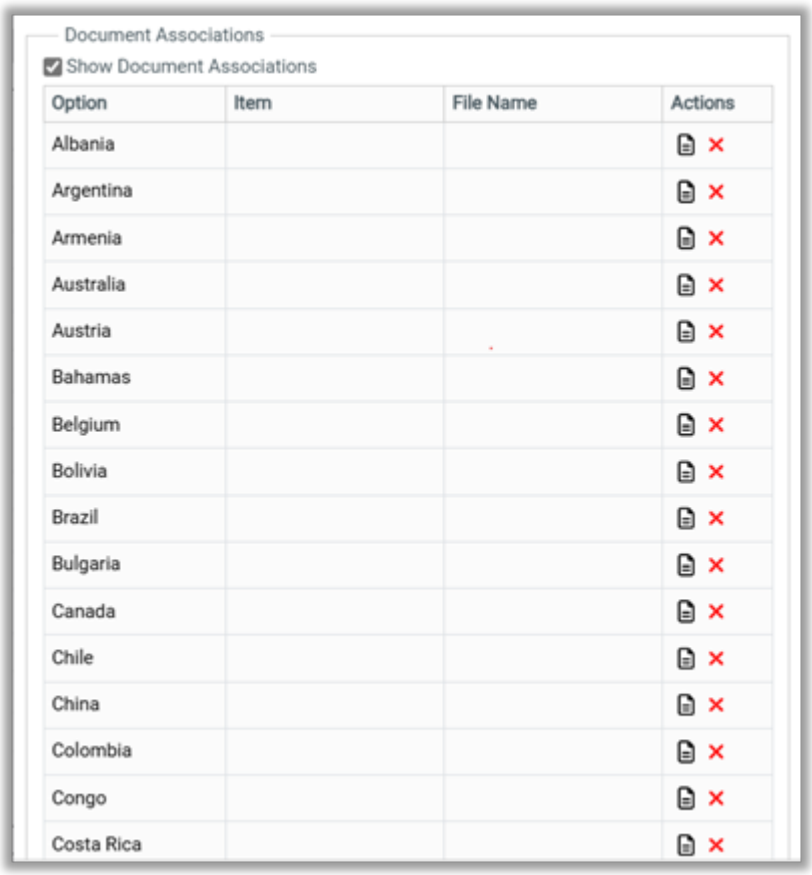
List control with Radio Group Control Style:

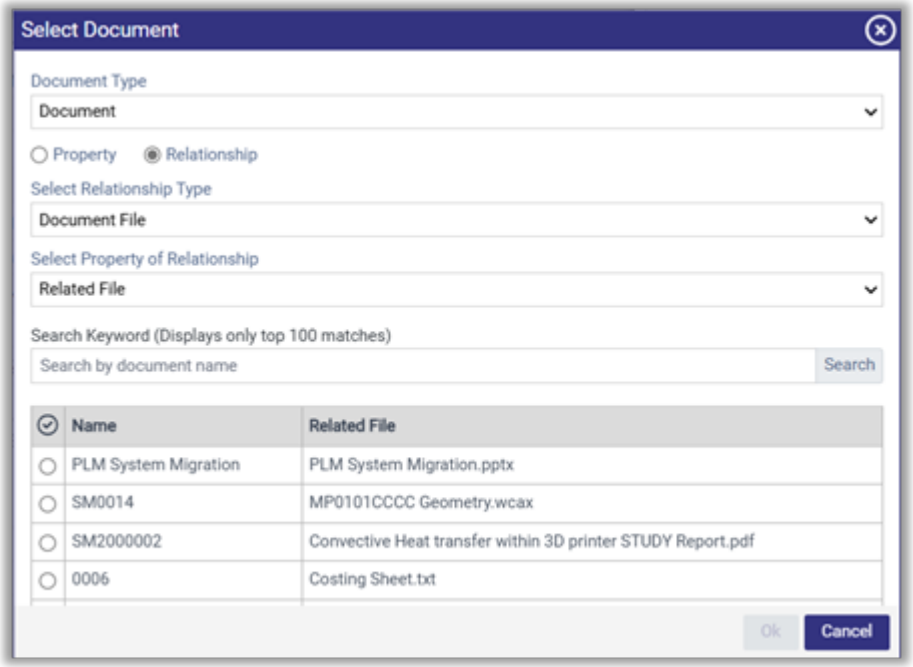


Multi List control with Dropdown Control Style:



- **Selection Mode**: It shows whether the current List control is a Single Value or Multi Value List. Selection mode is shown as disabled; it shows the list type as defined in the ItemType definition.
- **Document Associations**: It allows you to associate a document with each option of the List. These associations will be used while generating word file based on what options have been selected from the List. If this List property is used as ContentControl in the word template, ContentControl will be replaced with the documents selected through List options.

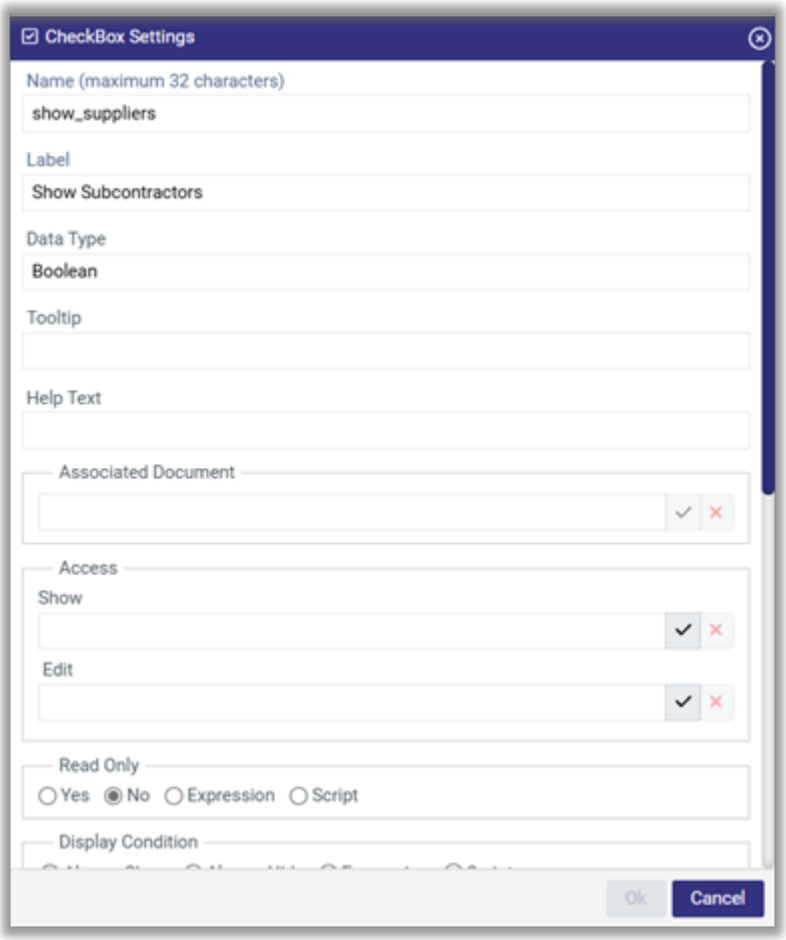




- **Access**: It allows you to define roles which can access control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, property associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If the returned value is false, the control will be hidden.
- **Validations**: Validation rules work on List control similar to Text control. Please refer to Validation section of the Text control for complete details.
- **Conditional Formatting**: Conditional Formatting works on List control similar to Text control. Please refer to the Conditional Formatting section of the Text control for complete details.
- **Reset Dependency**: It allows you reset value of the control to the default if any of the controls defined in “Reset Dependency” have their value changed at runtime.
- **Font**: It allows you to set the required font and size for showing list options and the selected value. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **Margin**: It allows you to set required top, right, bottom, left margins with respect to adjacent controls. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the value in the control is loaded. Please refer to the Template Customization section for more details.
- **On Change**: OnChange event allows you to write custom logic that will be executed after the value in the control is updated. Please refer to the Template Customization section for more details.

Checkbox Control

Checkbox control can be used to display properties of datatype Boolean from the associated ItemType.



- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show label on the control. It can be localized by defined Languages and Locales inside innovator.
- **Data Type**: It shows the type of the property as defined in the ItemType definition. In case of Checkbox control it will always be Boolean, and it is shown as disabled.
- **Tooltip**: Tooltip will be shown when you hover the mouse on the information icon on the form field. It is derived from the property bound to the control.
- **Help Text**: Help Text flyout will be shown when you click on the information icon on the form field label. It is derived from the property bound to the control.
- **Associated Document**: It allows you to associate a document with the CheckBox control. This associated document will be used while generating word file if this CheckBox is selected. If this Boolean property is used as ContentControl in the word

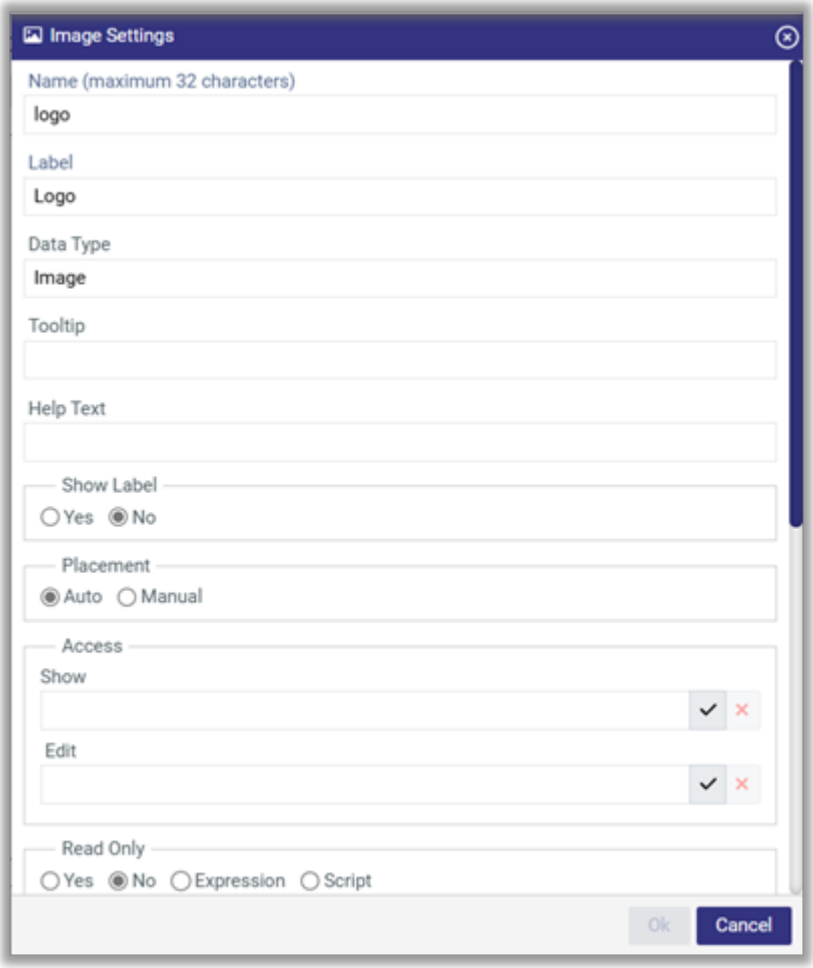
template, ContentControl will be replaced with the document associated with the CheckBox Control.

- **Access**: It allows you to define roles which can access the control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, property associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If the returned value is false, the control will be hidden.
- **Validations**: Validation rules work on Checkbox control similar to Text control. Please refer to Validation section of the Text control for complete details.
- **Reset Dependency**: It allows you reset value of the control to the default if any of the controls defined in “Reset Dependency” have their value changed at runtime.
- **Font**: It allows you to set the required font and size for the label of the checkbox.
- **Margin**: It allows you to set required top, right, bottom, left margins with respect to adjacent controls.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the value in the control is loaded. Please refer to the Template Customization section for more details.
- **On Change**: OnChange event allows you to write custom logic that will be executed after the value in the control is updated. Please refer to the Template Customization section for more details.

Image Control

Image control is used to display properties of datatype Image from the associated ItemType.

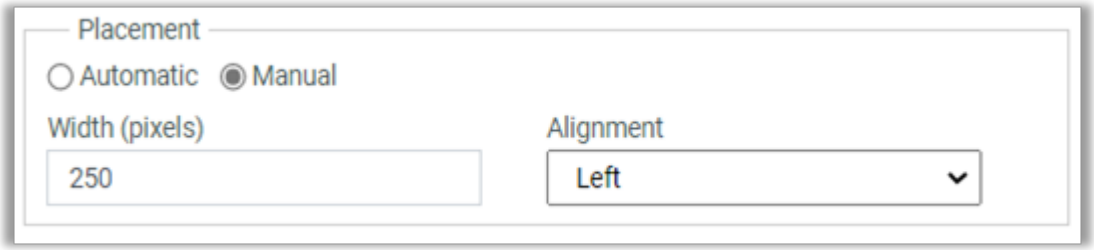
Image control will display the picture stored in the vault. It can also load the picture stored on the server folder by setting default value of image property as “../images/ImageName.svg” on ItemType definition.



- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show label on the control. It can be localized by defined Languages and Locales inside innovator.
- **Data Type**: It shows the type of the property as defined in the ItemType definition. In case of Image control, it will always be Image, and it is shown as disabled.
- **Tooltip**: Tooltip will be shown when you hover the mouse on the information icon on the form field. It is derived from the property bound to the control.

- **Help Text**: Help Text flyout will be shown when you click on the information icon on the form field label. It is derived from the property bound to the control.
- **Show Label**: If show label is selected as Yes, image will be shown with a label at top-left corner.

- **Placement**: It allows you to place the picture in the available space. By selecting placement as Automatic, picture will be scaled to available space without compromising aspect ratio. With Manual option, by setting width and alignment of the picture, height is computed with the help of aspect ratio.



- **Access**: It allows you to define roles which can access the control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, property associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If the returned value is false, the control will be hidden.
- **Reset Dependency**: It allows you reset value of the control to the default if any of the controls defined in “Reset Dependency” have their value changed at runtime.

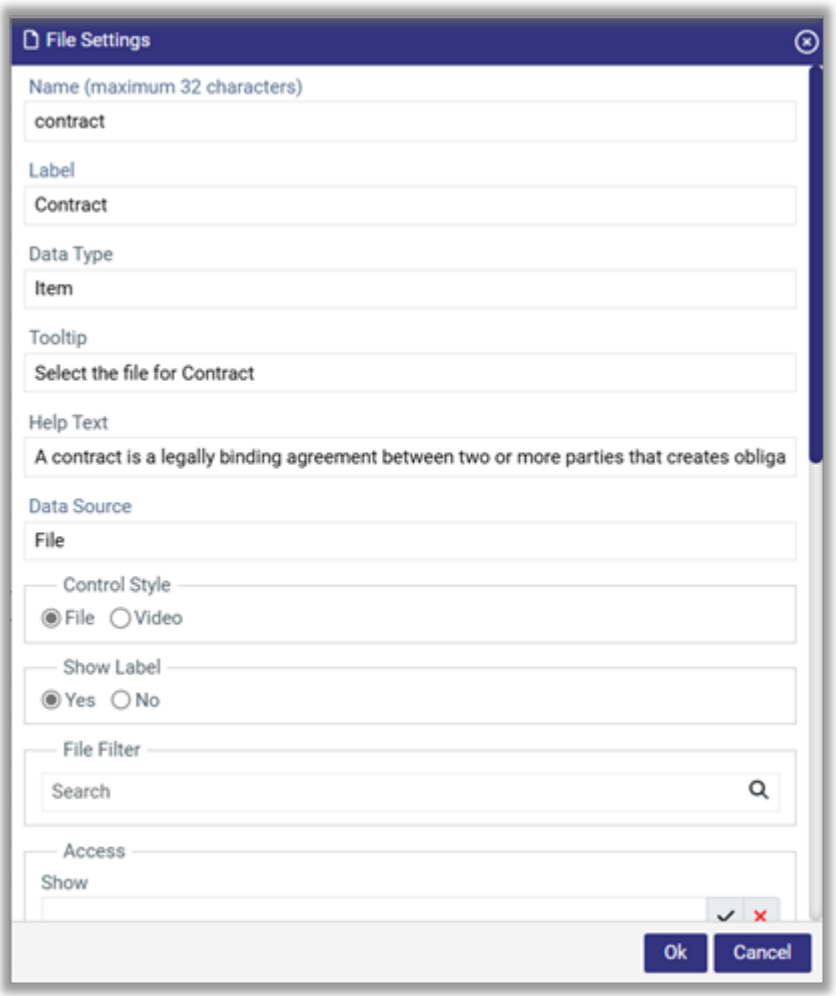
- **Validations**: Validation rules work on Image control similar to Text control. Please refer to Validations section of the Text control for complete details.
- **Margin**: It allows you to set required top, right, bottom, left margins with respect to adjacent controls.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the value in the control is loaded. Please refer to the Template Customization section for more details.
- **On Change**: OnChange event allows you to write custom logic that will be executed after the value in the control is updated. Please refer to the Template Customization section for more details.

Runtime View:

- **Upload**: By selecting this icon, you can upload picture from the local directory. Upload icon is enabled only when the item is in edit mode.
- **Download**: By selecting this icon, you can download the picture to a local directory. Download icon is always enabled if the image control has picture.
- **Capture**: By selecting this icon, you can capture the picture using device's camera. If multiple cameras are present, it will show you options to select specific camera.
- **Delete**: By selecting this icon, you can delete the picture shown in the image control (browser cache). Picture will be deleted from server only when the item is saved. Delete icon is enabled only when the item is in edit mode.

File Control

File control is used to display properties of datatype File from the associated ItemType. It can also show the file stored on the server by setting default value of file property with id of the file item on ItemType definition.



- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show label on the control. It can be localized by defined Languages and Locales inside innovator.
- **Data Type**: It shows the type of the property as defined in the ItemType definition. In case of File control, it will always be Item, and it is shown as disabled.
- **Tooltip**: Tooltip will be shown when you hover the mouse on the information icon on the form field. It is derived from the property bound to the control.

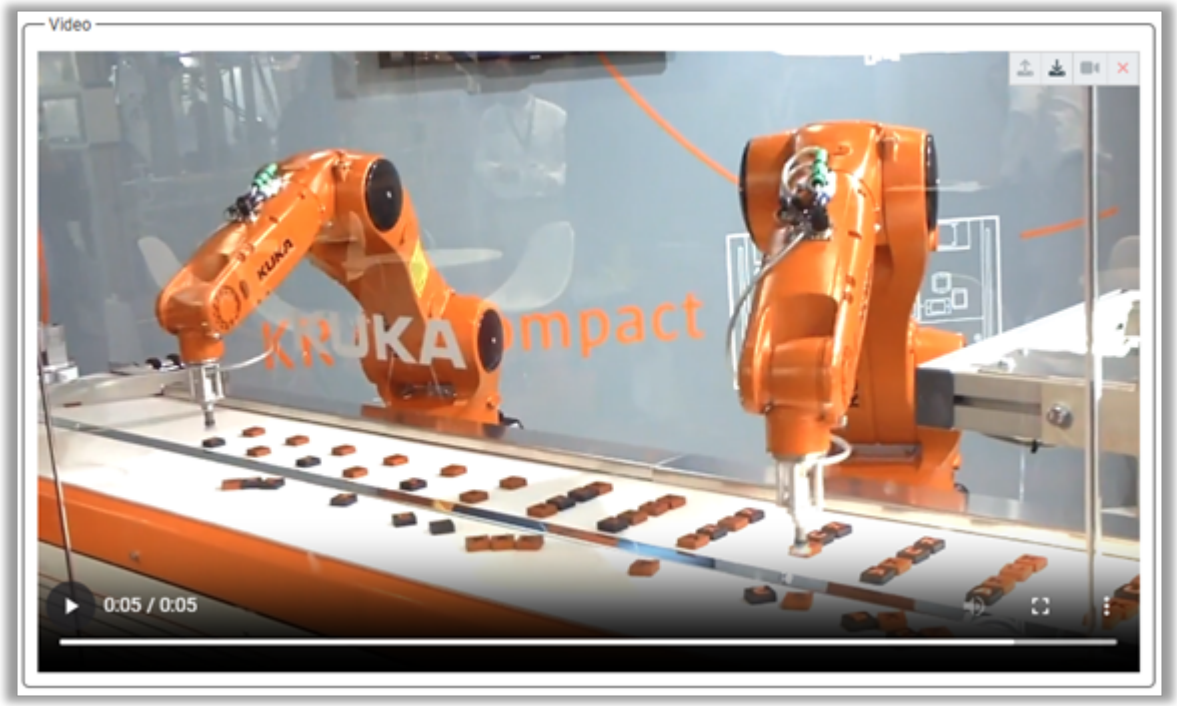
- **Help Text**: Help Text flyout will be shown when you click on the information icon on the form field label. It is derived from the property bound to the control.
- **Data Source**: It shows the data source of the property as defined in the ItemType definition. In case of File control, it will always be File, and it is shown as disabled.
- **Control Style**: File control can be rendered as File control for selecting file from the folder or Video Player to capture and play the video.
- **Show Label**: If show label is selected as Yes, video player will be shown with a label at top-left corner.
- **File Filter**: It allows you to set an allowed file format from the predefined list of options.
- **Access**: It allows you to define roles which can access the control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, property associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If the returned value is false, the control will be hidden.
- **Validations**: Validation rules work on File control similar to Text control. Please refer to Validations section of the Text control for complete details.
- **Reset Dependency**: It allows you reset value of the control to the default if any of the controls defined in “Reset Dependency” have their value changed at runtime.
- **Font**: It allows you to set the required font and size for the selected file of File control.
- **Margin**: It allows you to set required top, right, bottom, left margins with respect to adjacent controls.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the value in the control is loaded. Please refer to the Template Customization section for more details.
- **On Change**: OnChange event allows you to write custom logic that will be executed after the value in the control is updated. Please refer to the Template Customization section for more details.

Runtime View:

Control Type: File

- **Upload**: By selecting this icon, you can upload file from the local directory. Upload icon is enabled only when the item is in edit mode.
- **Download**: By selecting this icon, you can download the file to a local directory. Download icon is always enabled if the file control has value.
- **Delete**: By selecting this icon, you can delete the file shown in the file control (browser cache). File will be deleted from server only when the item is saved. Delete icon is enabled only when the item is in edit mode.

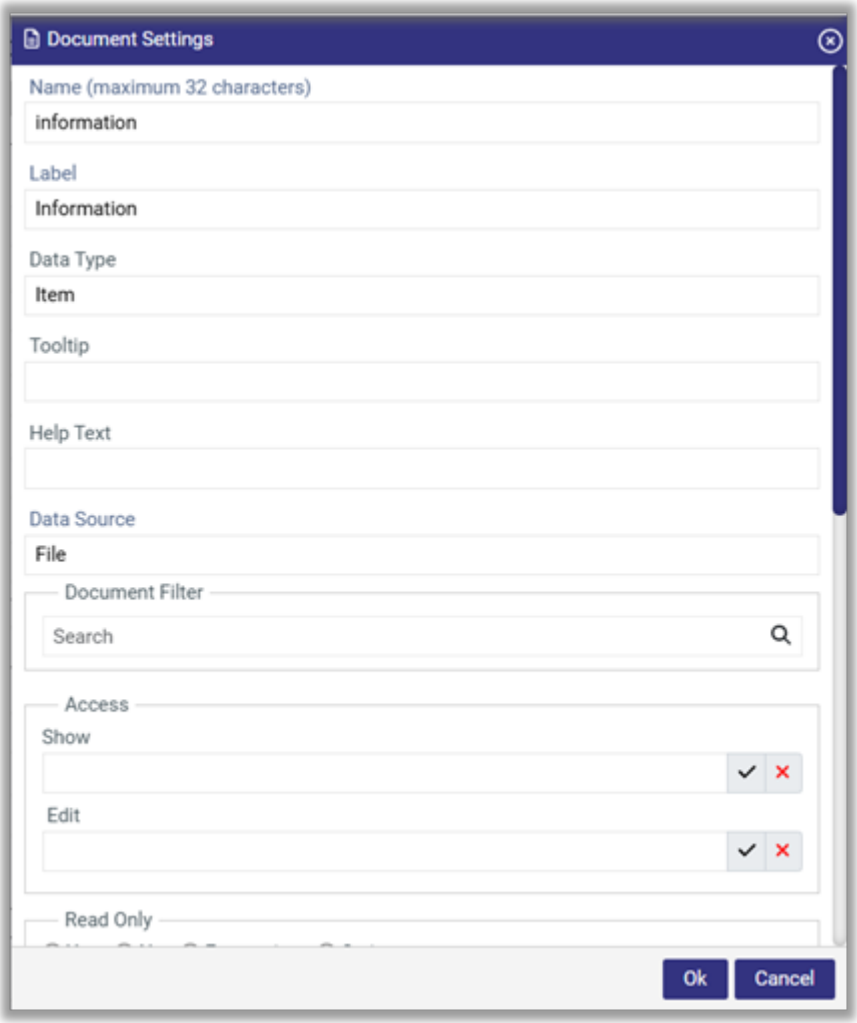
Control Type: Video



- **Capture**: By selecting this icon, you can capture the video using device's camera. If multiple cameras are present, it will show you options to select specific camera.

Document Control

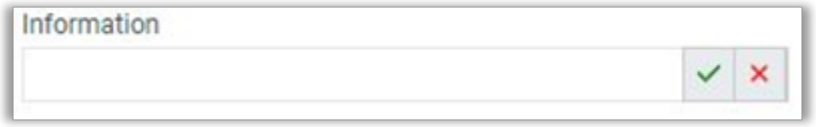
Document control is used to allow you to select the document which is stored in Innovator. Upon clicking on tick icon of the control, Select Document dialog will be shown. After entering search criteria, Select Document dialog shows all the files of type Word, PDF and Image. Selected document is stored on the context item as file property. Document control is useful in case you want to generate Word/PDF document from the context item.



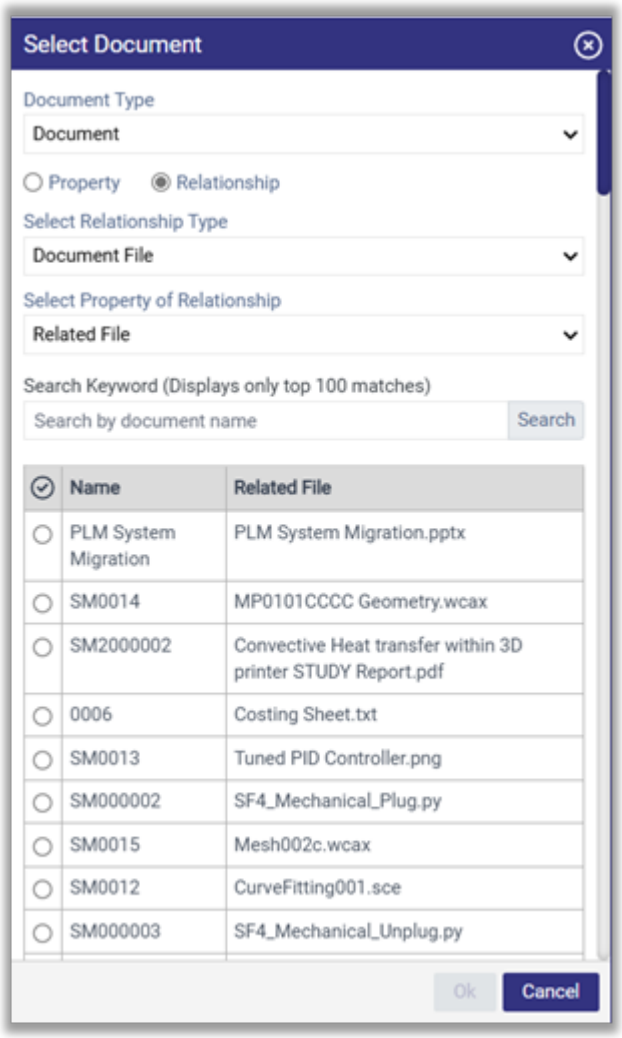
- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show label on the control. It can be localized by defined Languages and Locales inside innovator.
- **Data Type**: It shows the type of the property as defined in the ItemType definition. In case of Document control, it will always be Item, and it is shown as disabled.

- **Tooltip**: Tooltip will be shown when you hover the mouse on the information icon on the form field. It is derived from the property bound to the control.
- **Help Text**: Help Text flyout will be shown when you click on the information icon on the form field label. It is derived from the property bound to the control.
- **Data Source**: It shows source type of the property as defined in the ItemType definition. In case of Document control, it will always be File, and it is shown as disabled.
- **Document Filter**: It allows you to select the document types that you want to see in the search results. Allowed document types are Image, PDF, Word, Presentation.
- **Access**: It allows you to define roles which can access control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, property associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If the returned value is false, the control will be hidden.
- **Validations**: Validation rules work on Document control similar to Text control. Please refer to Validations section of the Text control for complete details.
- **Reset Dependency**: It allows you reset value of the control to the default if any of the controls defined in “Reset Dependency” have their value changed at runtime.
- **Font**: It allows you to set the required font and size for the selected document of Document control.
- **Margin**: It allows you to set required top, right, bottom, left margins with respect to adjacent controls.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the value in the control is loaded. Please refer to the Template Customization section for more details.
- **On Change**: OnChange event allows you to write custom logic that will be executed after the value in the control is updated. Please refer to the Template Customization section for more details.

Runtime View:

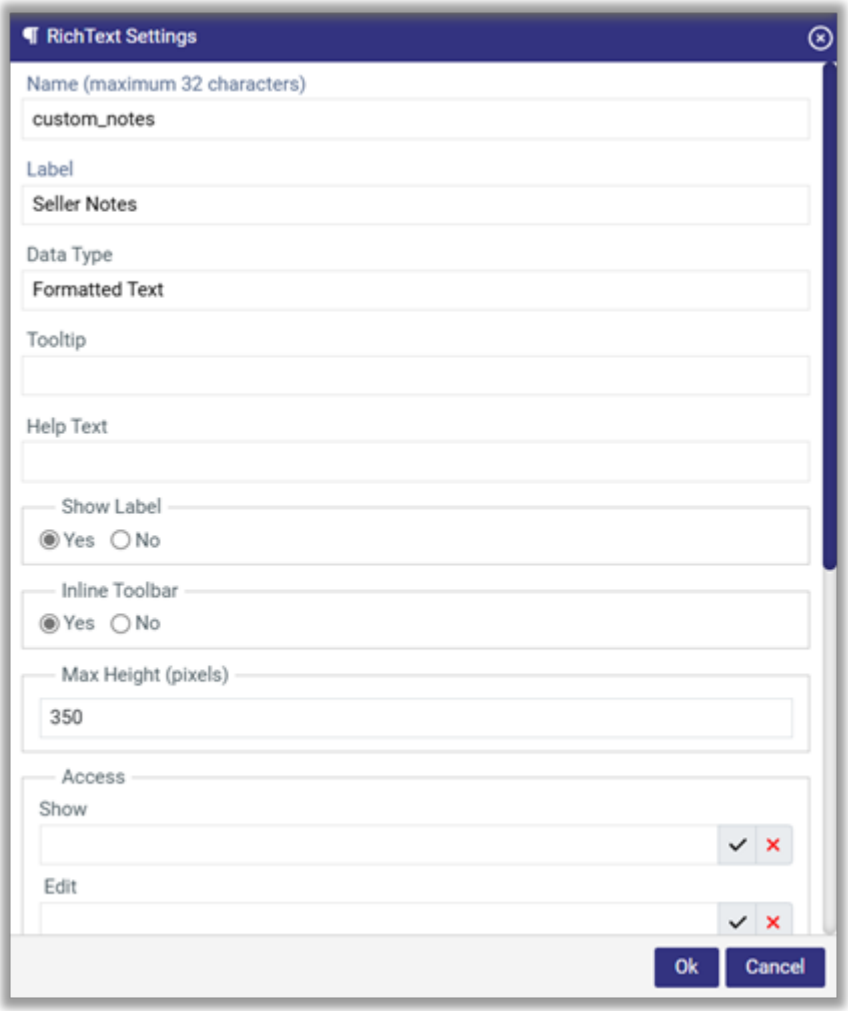


Select Document Flyout:



RichText Control

RichText control is used to display properties of datatype FormattedText from the associated ItemType. In edit mode, RichText control shows toolbar, using it you can format the text with bold, italic, font, color etc. It also allows you to insert items like images, hyperlinks and videos, upon saving all these items are embedded inside FormattedText on the property.



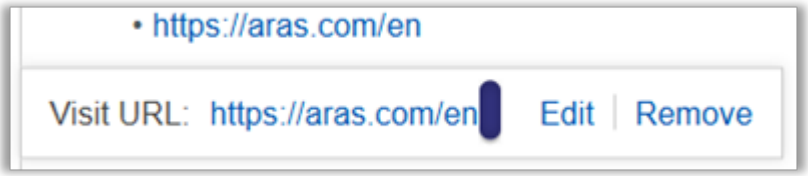
- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show label on the control. It can be localized by defined Languages and Locales inside innovator.
- **Data Type**: It shows the type of the property as defined in the ItemType definition. In case of RichText control, it will always be FormattedText, and it is shown as disabled.
- **Tooltip**: Tooltip will be shown when you hover the mouse on the information icon on the form field. It is derived from the property bound to the control.

- **Help Text**: Help Text flyout will be shown when you click on the information icon on the form field label. It is derived from the property bound to the control.
- **Show label**: If show label is selected as Yes, RichText control will be shown with a label at top-left corner.
- **Inline Toolbar**: If an inline toolbar is selected as Yes, toolbar will be shown within RichText control. Otherwise it will be shown in the popup after selecting the tick icon.
- **Max Height**: It is an optional setting, if you don't set any value for this, it will be shown with the height that fits the content to avoid scrollbar. By setting max height explicitly, RichText control will auto grow up to this height based on the content, if the content goes out of bounds, then it shows scrollbar.
- **Access**: It allows you to define roles which can access the control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, property associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like 'Yes', 'No', 'Expression', or 'Script'. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like 'Always Show', 'Always Hide', 'Expression', or 'Script'. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Reset Dependency**: It allows you reset value of the control to the default if any of the controls defined in "Reset Dependency" have their value changed at runtime.
- **Margin**: It allows you to set required top, right, bottom, left margins with respect to adjacent controls.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the value in the control is loaded. Please refer to the Template Customization section for more details.
- **On Change**: OnChange event allows you to write custom logic that will be executed after the value in the control is updated. Please refer to the Template Customization section for more details.

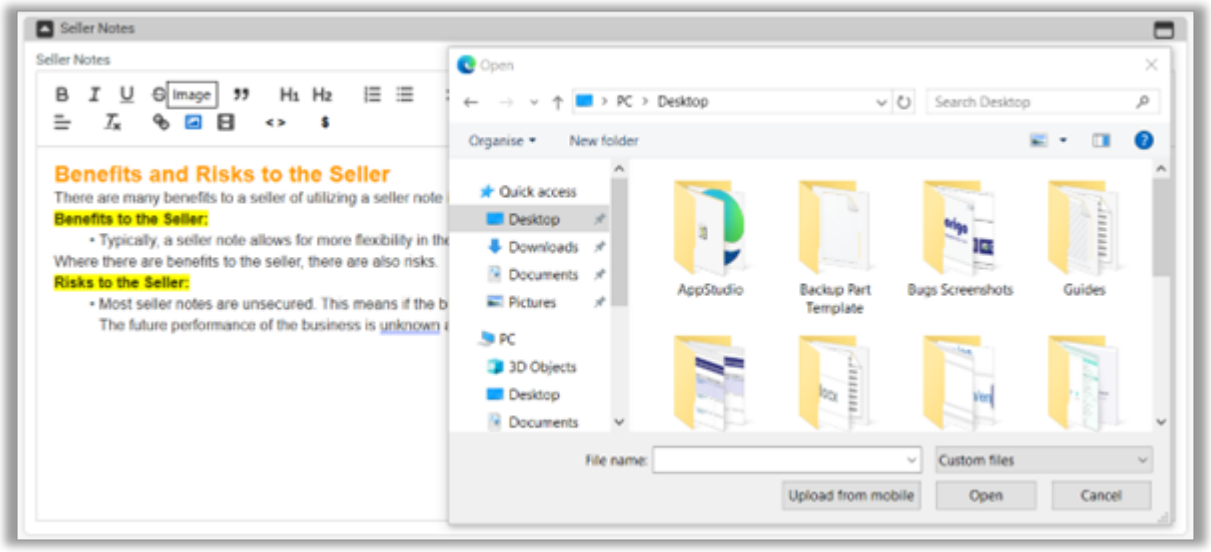
RichText Control Toolbar:

Hyperlink can be inserted in RichText control by highlighting the text and clicking on the null icon from toolbar. Then it shows pop-up as shown below to set link for the selected text.

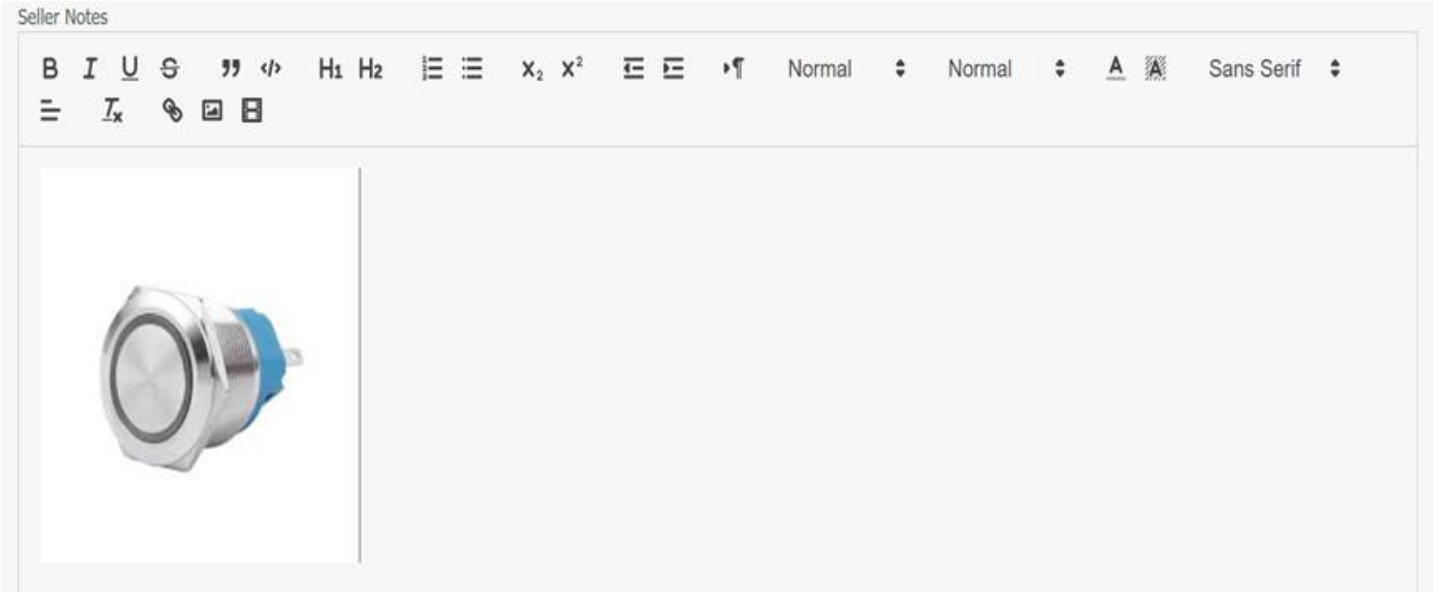
By clicking on the linked text, it shows the options to edit the current link or to remove link all together.



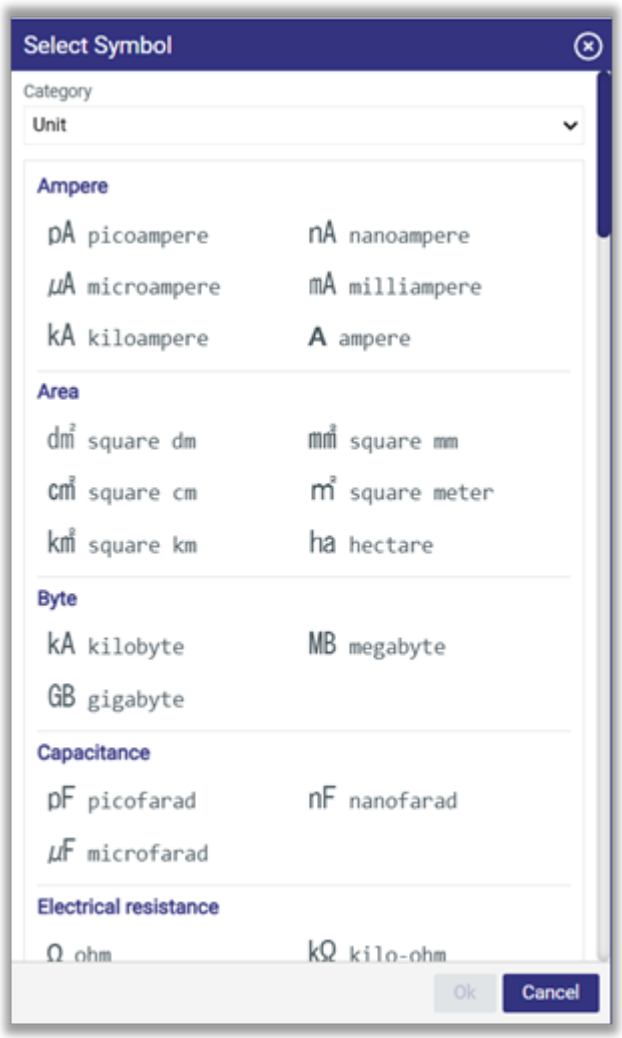
Picture can be inserted in RichText control at the current position of the cursor by selecting null icon from the toolbar. Then file browser dialog comes up in order to select the picture from the local directory.



The selected picture is displayed in the RichText control at the current cursor position.

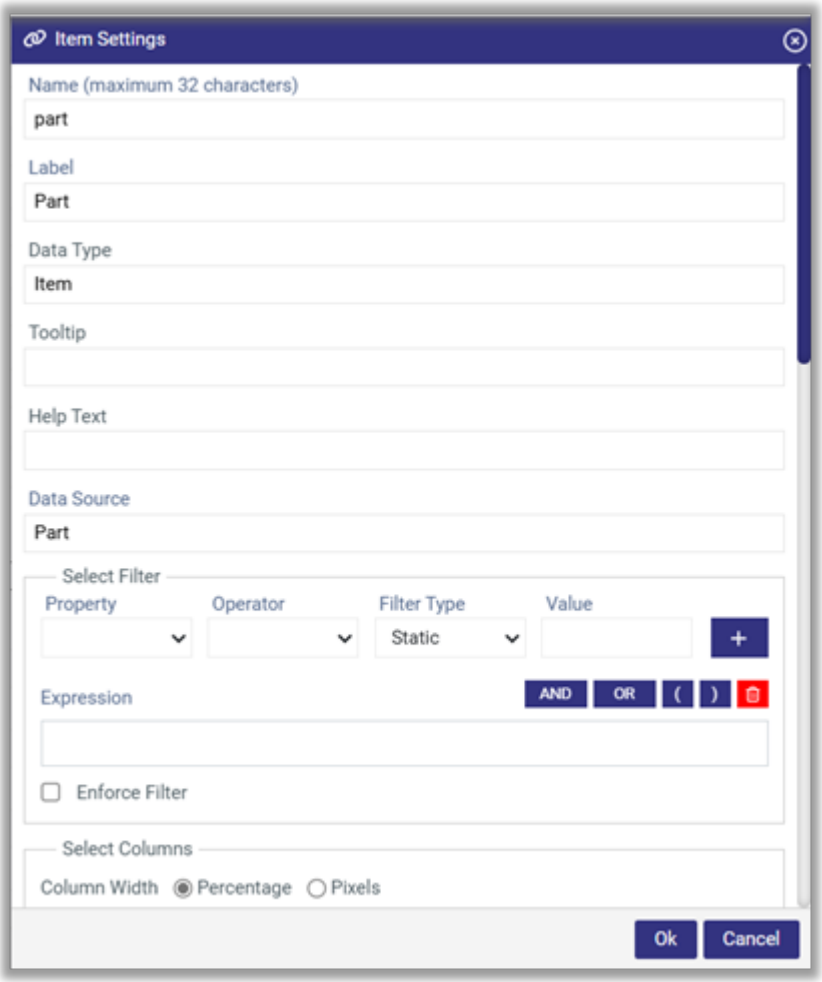


Raw HTML command from the toolbar will allow you to set any custom HTML as content for the RichText control. Selecting Insert Symbol command will show Select Symbol flyout, for inserting a symbol at the cursor position.



Item Control

Item control is used to display properties of datatype Item from the associated ItemType. Using this control, you can select existing item of the property's Data Source ItemType through Select Dialog.

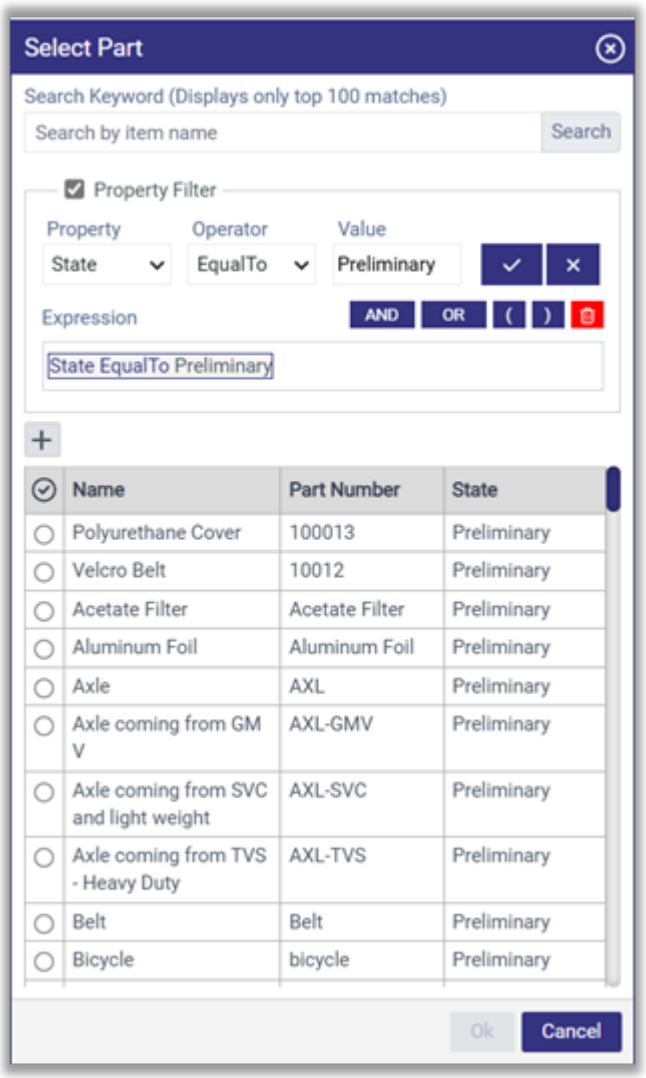


- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show label on the control. It can be localized by defined Languages and Locales inside innovator.
- **Data Type**: It shows the type of the property as defined in the ItemType definition. In case of Item control, it will always be Item, and it is shown as disabled.
- **Tooltip**: Tooltip will be shown when you hover the mouse on the information icon on the form field. It is derived from the property bound to the control.
- **Help Text**: Help Text flyout will be shown when you click on the information icon on the form field label. It is derived from the property bound to the control.
- **Data Source**: It shows the type of the Item that can be selected as defined in the ItemType definition. It is shown as disabled.

- **Select Filter**: It allows you to select filter criteria based on the properties of the Related ItemType. Defined filter is always applied with ‘And’ condition on the Selected Items dialog along with the Search Keyword. If FilterType is Static, for the expression you need to provide constant value. If FilterType is Dynamic, for the expression you need to provide other property name as value, with this at runtime it will pull the value from the selected property and send it in the filter request to the server. As an admin, you can also enforce the filter, in that case end-user can see the applied filter on the Select Dialog but can’t change the filter criteria. If enforce filter is not set, end-user can change the filter criteria at the runtime.
- **Select Columns**: It allows you to define columns that you want to show in the Select Dialog. Column widths can be given in percentage or pixels modes. In percentage mode, you never see horizontal scrollbar, column widths will be adjusted to the available dialog width. But in pixels mode, based on the given column widths you may see horizontal scrollbar in the dialog. Flyout width in percentage, allows you to set width of the flyout shown for selecting the item. By selecting Show Add command ‘Yes’, Select Item flyout with show toolbar with Add Item command on the table. Using this command, new item can be created on the fly as part of the select item operation.
- **Access**: It allows you to define roles which can access control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, property associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Validations**: Validation rules work on Item control similar to Text control. Please refer to Validations section of the Text control for complete details.
- **Reset Dependency**: It allows you reset value of the control to the default if any of the controls defined in “Reset Dependency” have their value changed at runtime.
- **Font**: It allows you to set the required font, size and color for the value of Item control.
- **Margin**: It allows you to set required top, right, bottom, left margins with respect to adjacent controls.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the value in the control is loaded. Please refer to the Template Customization section for more details.
- **On Change**: OnChange event allows you to write custom logic that will be executed after the value in the control is updated. Please refer to the Template Customization section for more details.
- **On Select Filter**: On Select Filter event will be launched after hitting the search button in item selection flyout. Admin can configure a custom script to modify entered filter expression or assign entirely new filter expression string to the eventData. EventData object is available in the scope of the event handler code. Please refer to the Template Customization section for more details.

Runtime View:

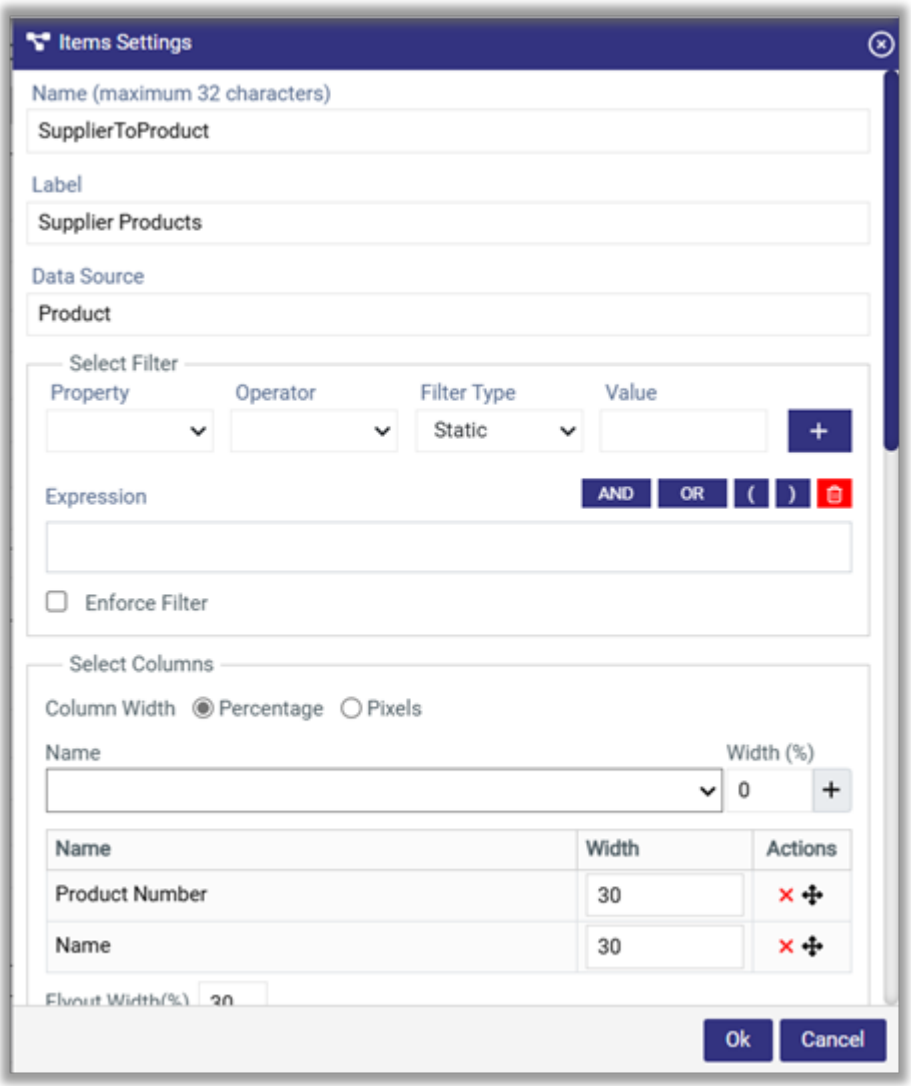
- **Select***:* By selecting tick icon, you will get ‘Select Item’ flyout from which you can search based on keyword/filter and select a specific item. By clicking on search after entering keyword or filter criteria, you will get a list of matched items, and you can select one of the items. Item control supports typeahead; by entering first three characters, it shows all matched items in the dropdown.



- **Remove**: By selecting remove icon, you can remove existing item. Item reference will be removed only when you save the context item.

Items Control

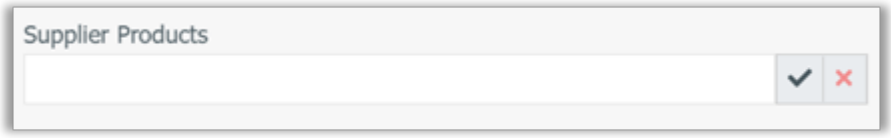
Items control is used to select and display list of items with their keyed-names from the associated ItemType. Using this control, you can select existing items of the property's Data Source ItemType through Select Dialog. Selected items will be saved as relationships upon saving the context item.



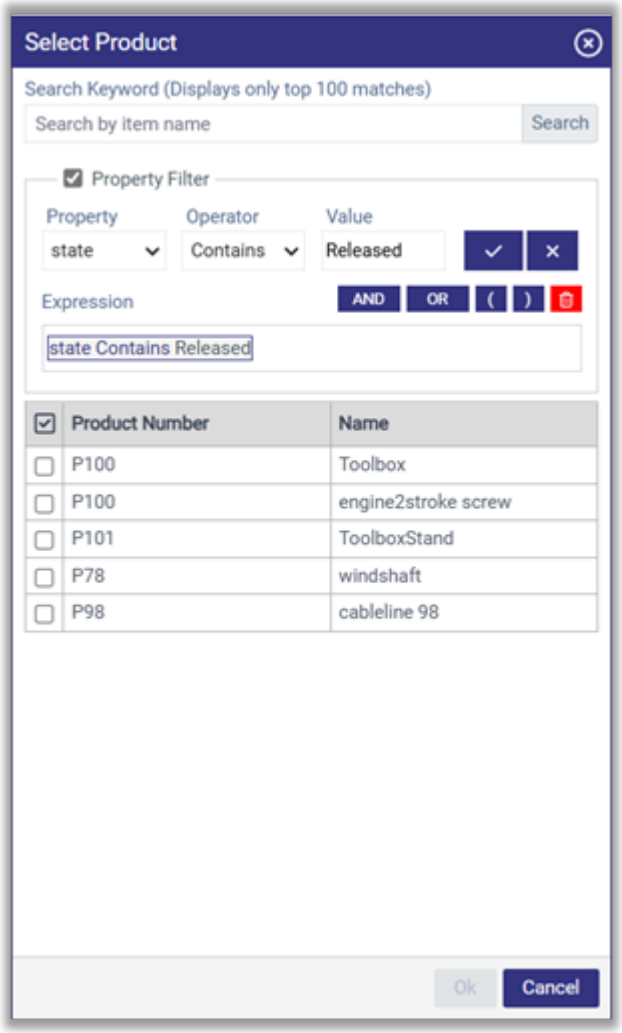
- **Name**: It is used to uniquely identify control inside the template. It is always the selected RelationshipType name.
- **Label**: It is used to show label on the control. It can be localized by defined Languages and Locales inside innovator. It always shows selected RelationshipType label.
- **Data Source**: It shows the type of the Item that can be selected as defined in the ItemType definition. It is shown as disabled.

- **Select Filter**: It allows you to select filter criteria based on the properties of the Related ItemType. Defined filter is always applied with ‘And’ condition on the Selected Items dialog along with the Search Keyword. If FilterType is Static, for the expression you need to provide constant value. If FilterType is Dynamic, for the expression you need to provide other property name as value, with this at runtime it will pull the value from the selected property and send it in the filter request to the server. As an admin, you can also enforce the filter, in that case end-user can see the applied filter on the Select Dialog but can’t change the filter criteria. If enforce filter is not set, end-user can change the filter criteria at the runtime.
- **Select Columns**: It allows you to define columns that you want to show in the Select Dialog. Column widths can be given in percentage or pixels modes. In percentage mode, you never see horizontal scrollbar, column widths will be adjusted to the available dialog width. But in pixels mode, based on the given column widths you may see horizontal scrollbar in the dialog. Flyout width in percentage, allows you to set width of the flyout shown for selecting the items. By selecting Show Add command ‘Yes’, Select Items flyout with show toolbar with Add Item command on the table. Using this command, new item can be created on the fly as part of the select items operation.
- **Access**: It allows you to define roles which can access the control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, property associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Validations**: Validation rules work on Items control similar to Text control. Please refer to Validations section of the Text control for complete details.
- **Reset Dependency**: It allows you reset value of the control to the default if any of the controls defined in “Reset Dependency” have their value changed at runtime.
- **Font**: It allows you to set the required font, size and color for the value of Items control.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the value in the control is loaded. Please refer to the Template Customization section for more details.
- **On Change**: OnChange event allows you to write custom logic that will be executed after the value in the control is updated. Please refer to the Template Customization section for more details.
- **On Select Filter**: On Select Filter event will be launched after hitting the search button in item selection flyout. Admin can configure a custom script to modify entered filter expression or assign entirely new filter expression string to the eventData. EventData object is available in the scope of the event handler code. Please refer to the Template Customization section for more details.

Runtime View:

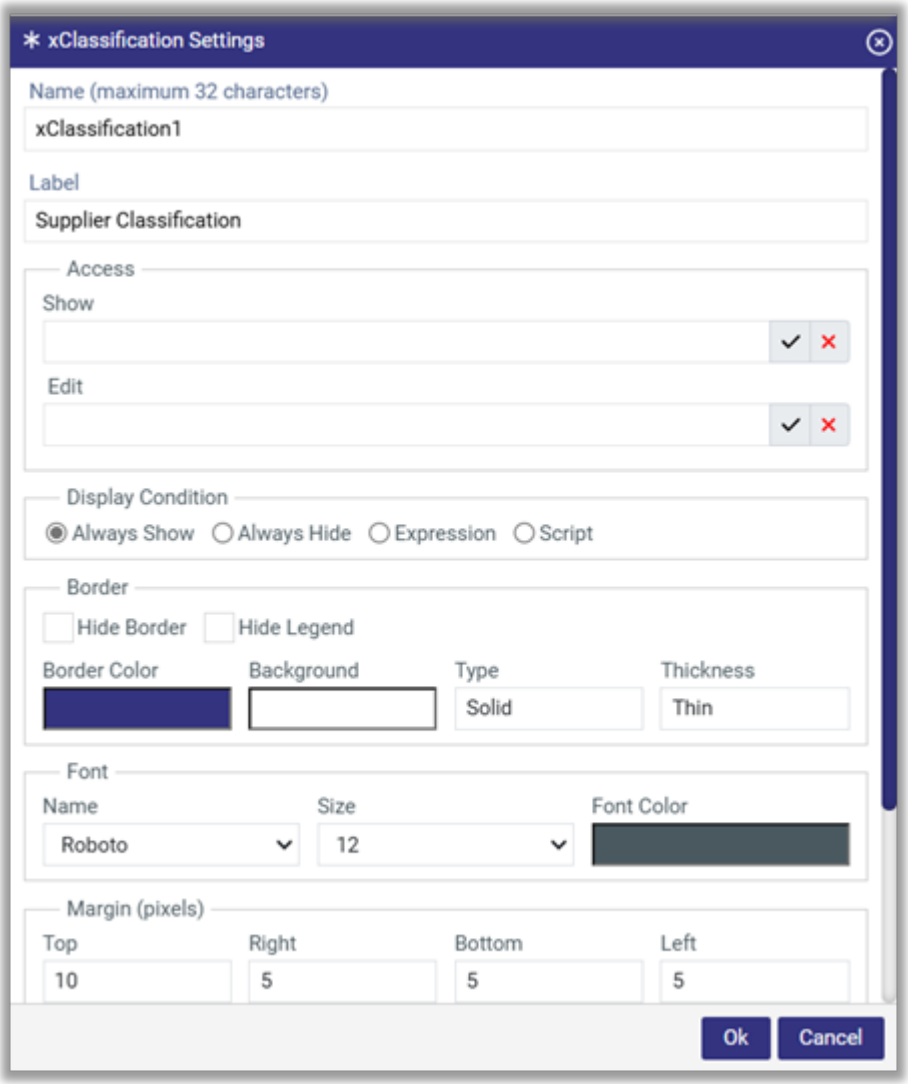


- **Select**: By selecting tick icon, you will get ‘Select Items’ flyout from which you can search based on keyword/filter and select list of items. By clicking on search after entering keyword or filter criteria, you will get a list of matched items, and you can select multiple items.



xClassification Control

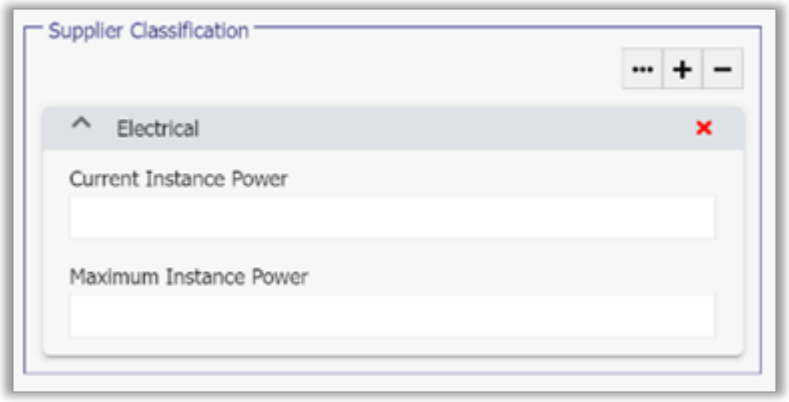
xClassification control is used to associate xClassification tree nodes and xProperties with the item. It works exactly same way as you work with regular Innovator forms.



- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title of the group. It can be localized by defined Languages and Locales inside innovator.
- **Access**: It allows you to define roles which can access control. Roles (in the form group identities) can be at Show and Edit level. When group is hidden because you do not have defined role, properties used inside this group are not even fetched from server.

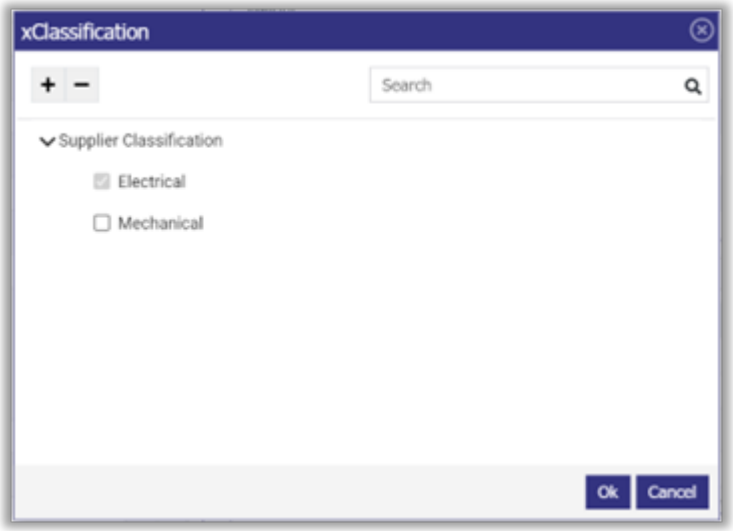
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Border**: It allows you to show or hide the border and legend around selected xClasses. In case if you choose to show the border, you can select the border style using Color, Background, Type and Thickness settings. If any of the settings are changed, it shows Reset icon on the border, using which you can reset the settings.
- **Font**: It allows you to set the required font and size on the individual fields of the classification properties.
- **Margin**: It allows you to set required top, right, bottom, left margins with respect to adjacent controls.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the value in the control is loaded. Please refer to the Template Customization section for more details.
- **On Change**: OnChange event allows you to write custom logic that will be executed after the value in the control is updated. Please refer to the Template Customization section for more details.

Runtime View:



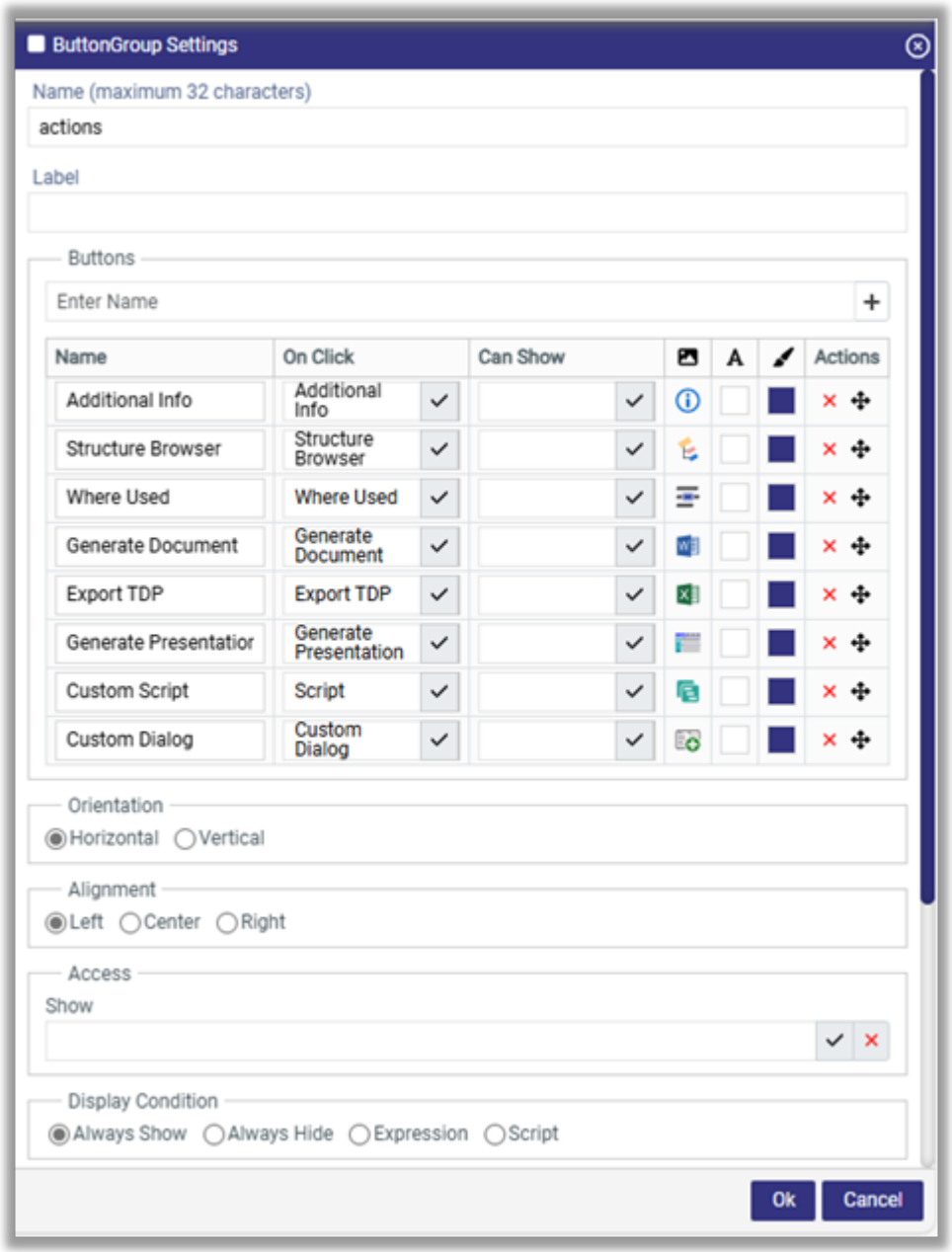
When you click on "…" from xClassification control runtime view, you will see xClassification tree selection dialog. Select specific subclass and click on Ok, you will see selected subclass along with its xProperties in the control.

- **Expand/Collapse**: Displays or hides all the subclasses in a multiple level hierarchy.
- **Search Filter**: Displays the classes and subclasses containing the characters typed into the filter box. For example, to only show subclasses that start with the keyword "El", enter El into the search field, and only those subclasses starting with El will be displayed.



ButtonGroup Control

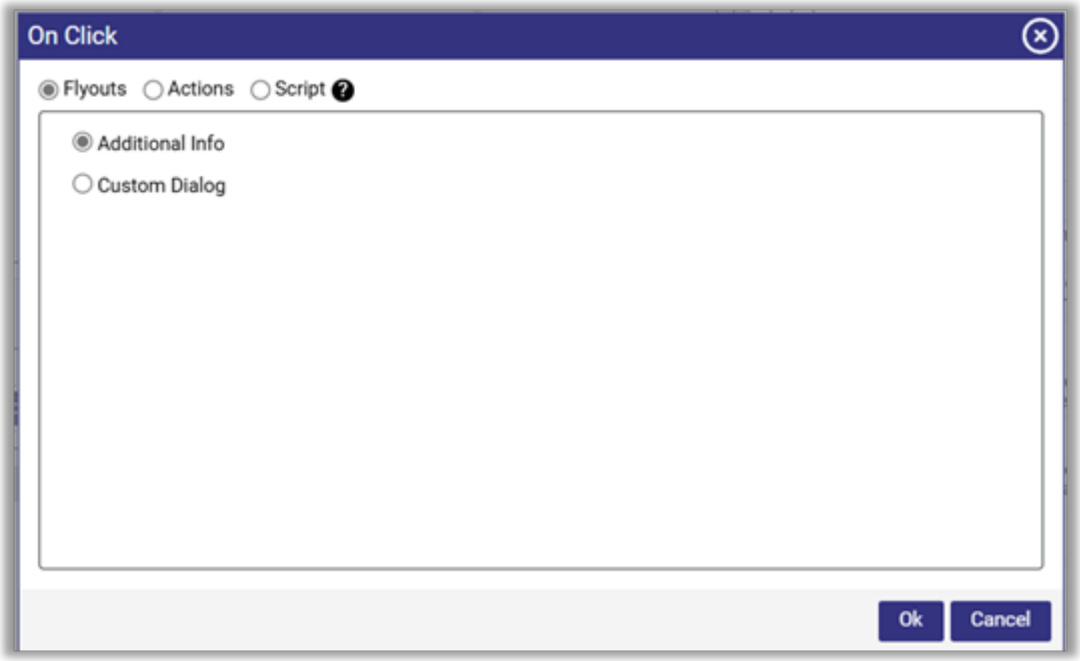
ButtonGroup control is used to display a group of buttons or a single button. Using button control you can implement custom action. Each button can be configured to show a dialog or a shortcut to the existing Innovator page. Button can also be configured to call a custom script.

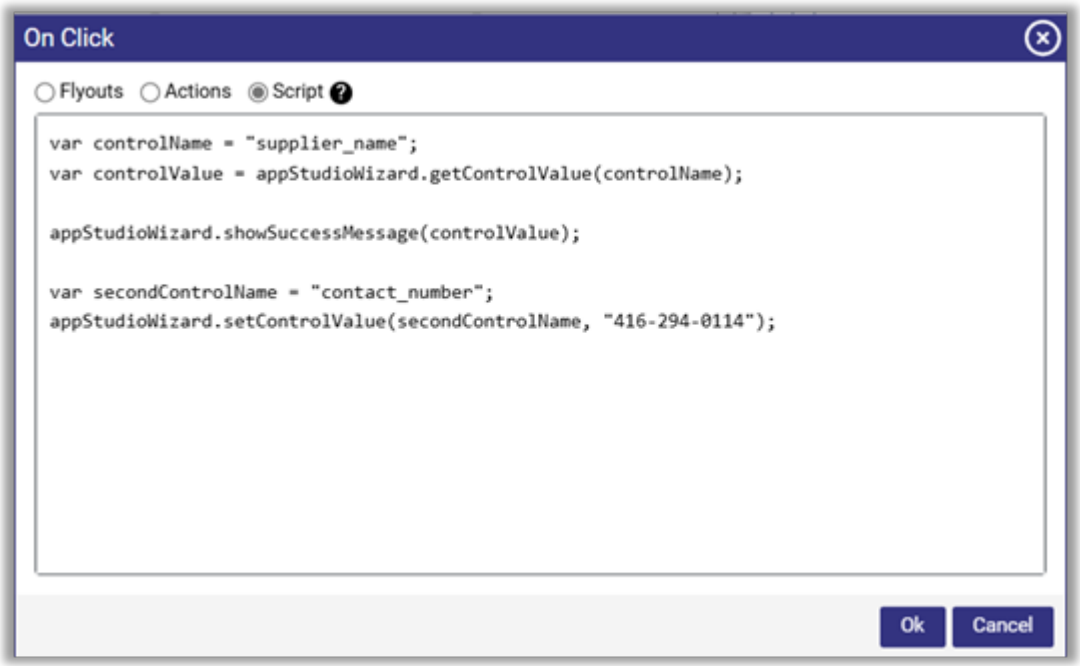


- **Name***:* It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the button group. It can be localized by defined Languages and Locales inside innovator.

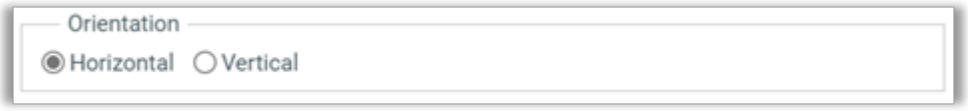
- **Buttons**: It allows you to define single or multiple buttons in the group. You can configure each button with a name and on-click handler. Each button can also be assigned with an icon, fore color, and background color.

  When you click on tick icon under OnClick column the following dialog will be shown. In the dialog, you can select one of the dialogs defined in the template that will be shown when user clicks on the button. You can also configure existing Innovator actions like Structure Browser, Where Used, Workflow Signoffs to be shown in Innovator tab for the context item. By selecting Script option in the dialog, you can associate custom script that contains code to interact with other controls in the Wizard. Script can also have code to interact with server by sending a request. For each button, you can configure icon, text and background colors. For each button, you can also define “Can Show” script that should return Boolean value. If returned value is false, button hidden. If it is null, button will be shown as disabled.

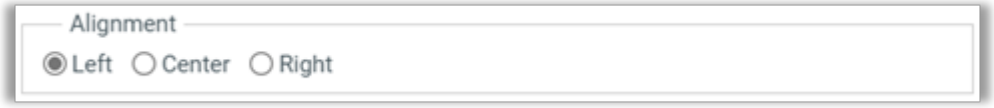




- **Orientation**: It allows you to show buttons either horizontally or vertically stacked within its parent group control.



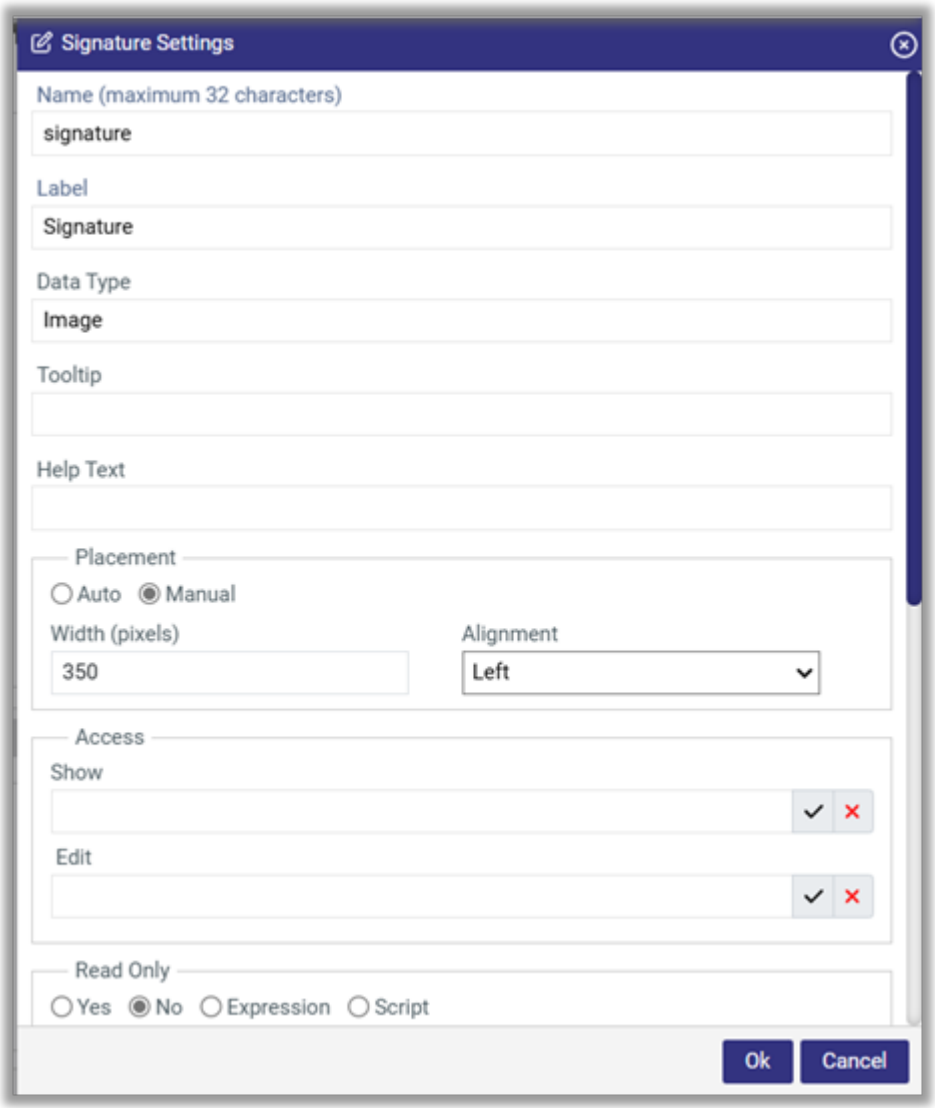
- **Alignment**: It allows you to show buttons aligned to left, center, or right in the available width of the parent group control.



- **Access**: It allows you to define roles which can access control. You can configure Roles (in the form group identities) to whom this control should be shown.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Border**: It allows you to show or hide the border and legend of the button group.
- **Font**: It allows you to set the required font and size for the button names.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.

Signature Control

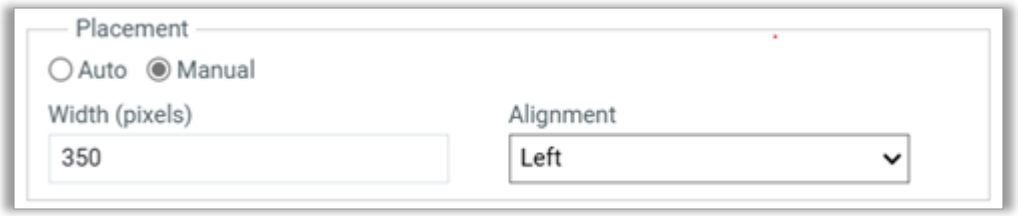
Signature control can be used to capture digital signature of you as image and can be saved to image property of the context item. Signature control shows signature pad; you can use mouse to draw your signature.



- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show label on the control. It can be localized by defined Languages and Locales inside innovator.
- **Data Type**: It shows the type of the property as defined in the ItemType definition. In case of Signature control it will always be Image, and it is shown as disabled.

- **Tooltip**: Tooltip will be shown when you hover the mouse on the information icon on the form field. It is derived from the property bound to the control.
- **Help Text**: Help Text flyout will be shown when you click on the information icon on the form field label. It is derived from the property bound to the control.

- **Placement**: It allows you to place the control in the available space. By selecting placement as Automatic, control will be scaled to available space. With Manual option, you can select alignment and width of the control.

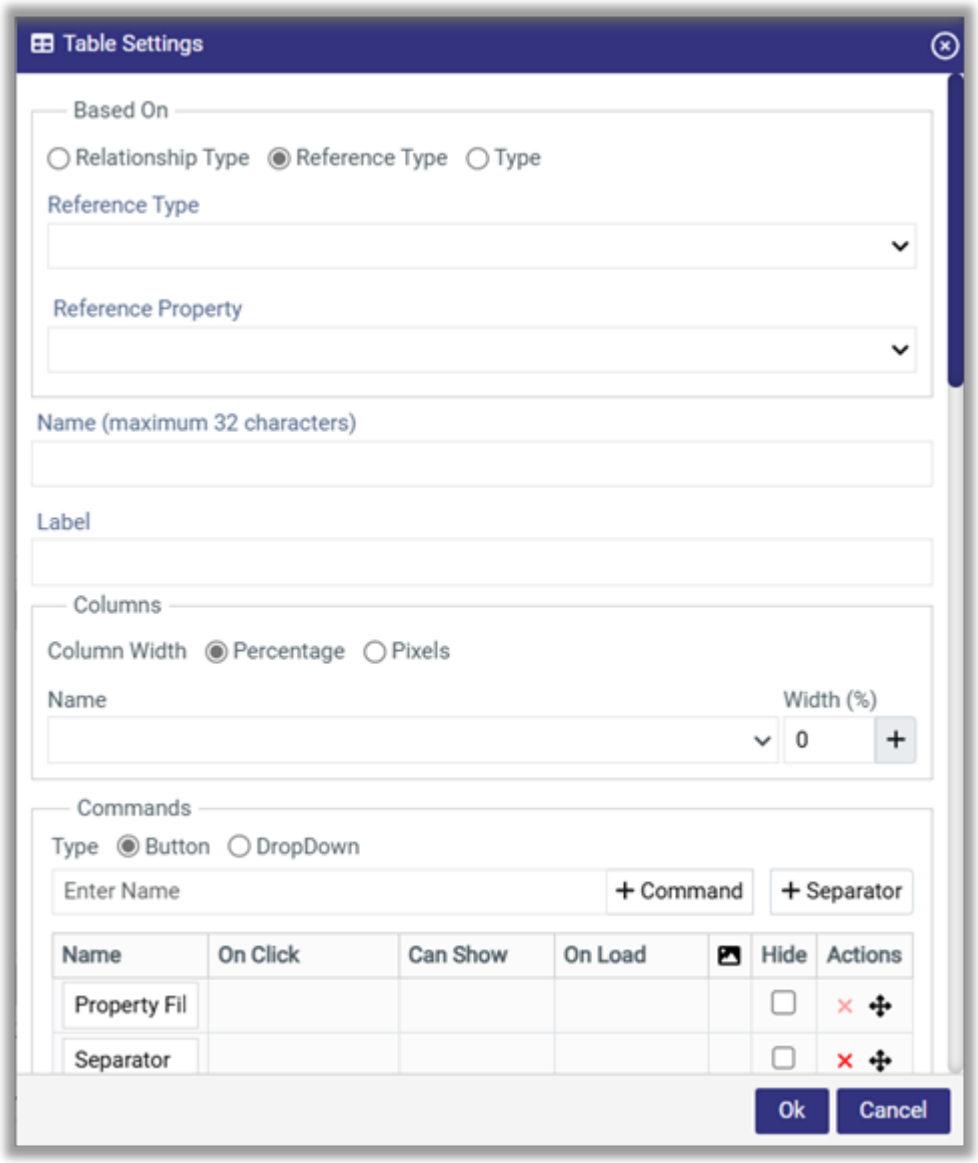


- **Access**: It allows you to define roles which can access the control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, property associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Validation**: Validation rules work on Signature control similar to Text control. Please refer to Validations section of the Text control for complete details.
- **Reset Dependency**: It allows you reset value of the control to the default if any of the controls defined in “Reset Dependency” have their value changed at runtime.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the value in the control is loaded. Please refer to the Template Customization section for more details.
- **On Change**: OnChange event allows you to write custom logic that will be executed after the value in the control is updated. Please refer to the Template Customization section for more details.

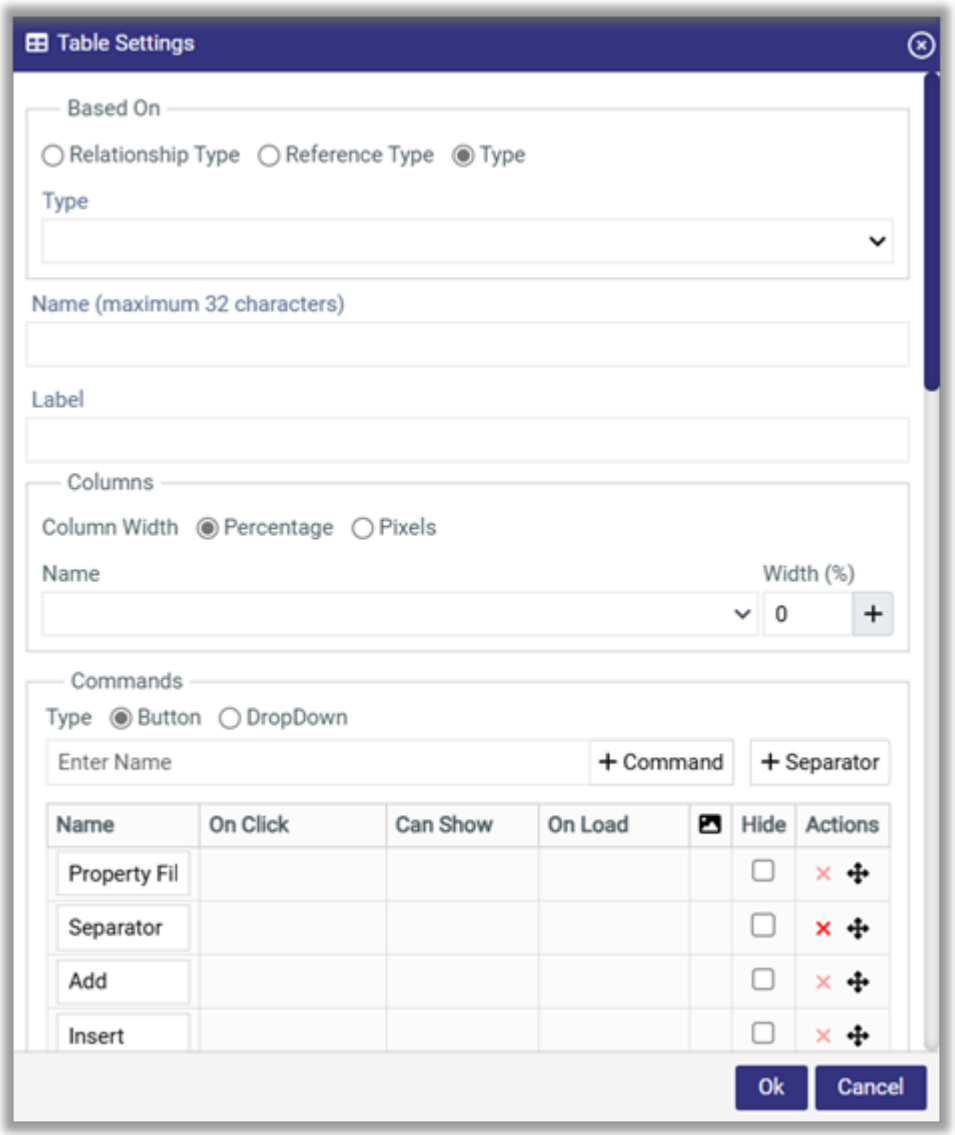
Table Control

Table control is used to show relationship or reference data of the context item. As part of adding table control to the form, Aras ProAppDesigner shows Settings flyout where you can configure the RelationshipType or ReferenceType that you want to associate with the table control. RelationshipType or ReferenceType can only be selected at the time of adding table control. If you need to change the associated RelationshipType or ReferenceType, then you need to delete table control and re-add it. Table control has multiple settings that controls its appearance and behavior at runtime.

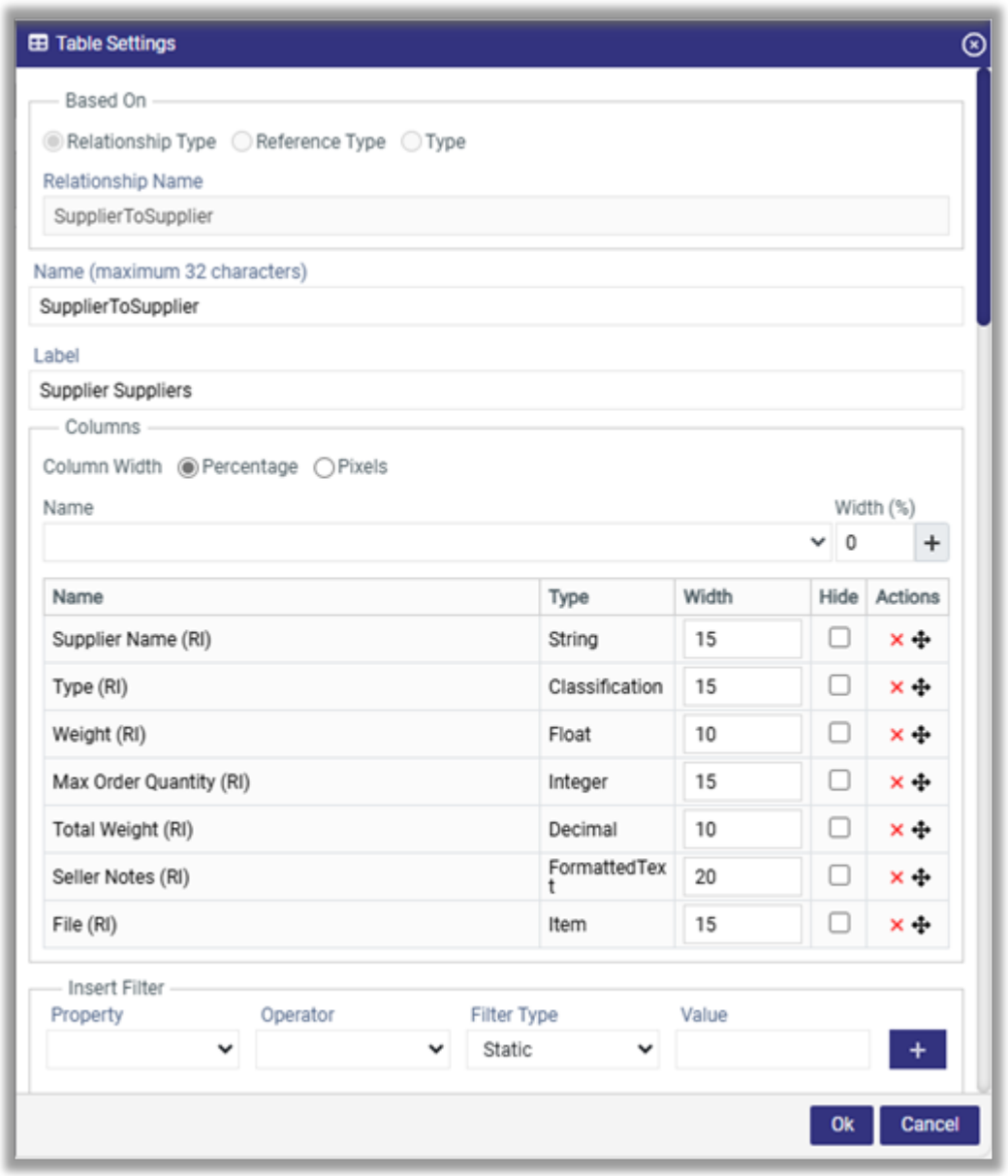
When ReferenceType option is selected for creating Table control, a dropdown will be shown with all ItemTypes on which context ItemType is defined as item property. If there are multiple item properties of the context ItemType on the selected ItemType, a dropdown will be shown to select the specific item property.



When Type option is selected for creating Table control, a dropdown will be shown with all ItemTypes available in the system. At runtime, table shows rows of the items from the configured ItemType. By default, it doesn't show any rows. In order to see the rows, you have to set the property filter and click on the Search icon. This table configuration can be used to show table in the flyout and let user select specific item after performing search with the filter.



You can also configure individual columns on the table control by selecting gear icon from the column. You can set Validation Rules, Default Value, Conditional Formatting, Read Only on the individual column by selecting gear icon from the column that opens Settings flyout. While defining expressions on the individual columns, you will only see properties from other columns of the same table control.

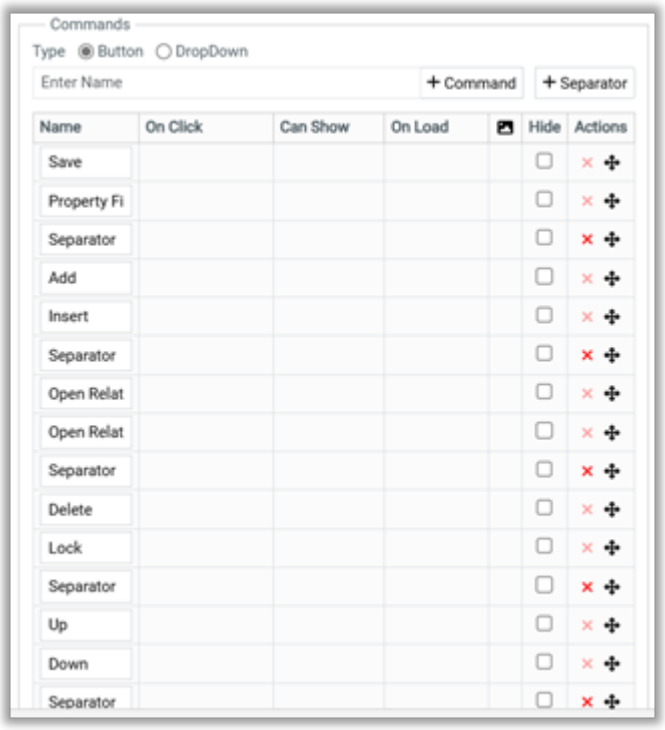


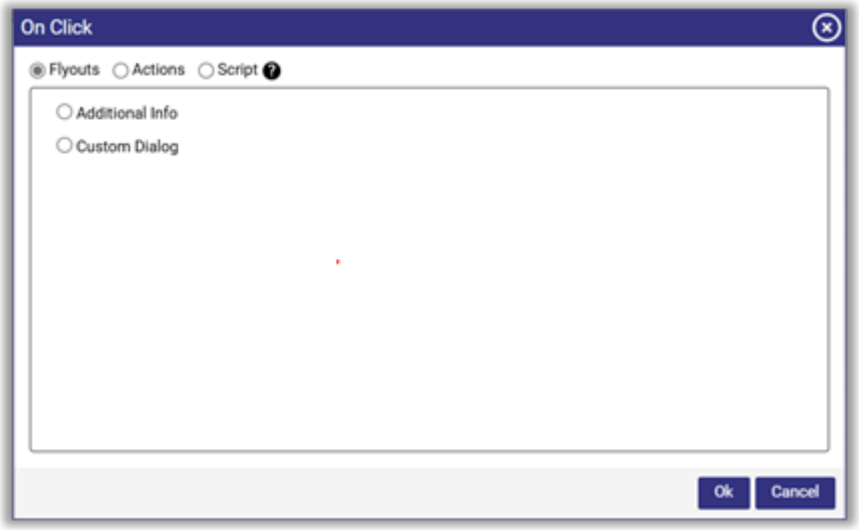
- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the table. It can be localized by defined Languages and Locales inside innovator.
- **Columns**: Using Columns settings section, you can add all the required columns to the table. Columns can be added based on the RelationshipType or RelatedItemType properties. Using Column Width option, you can set column widths in percentage or pixels. If you select Column Width as Percentage, you never see horizontal scrollbar, widths of the columns will get adjusted to the available total width of the table. If Column Width option is selected as Pixels, you will see horizontal scrollbar if the combined widths of all columns is more than available table width. In pixel mode, columns can be reordered and resized at runtime. As part of adding column, you need to select the property from either RelationshipType or RelatedItemType, to which you want to bind the column. Columns added with RelatedItemType properties will be shown with (RI) in the column name.

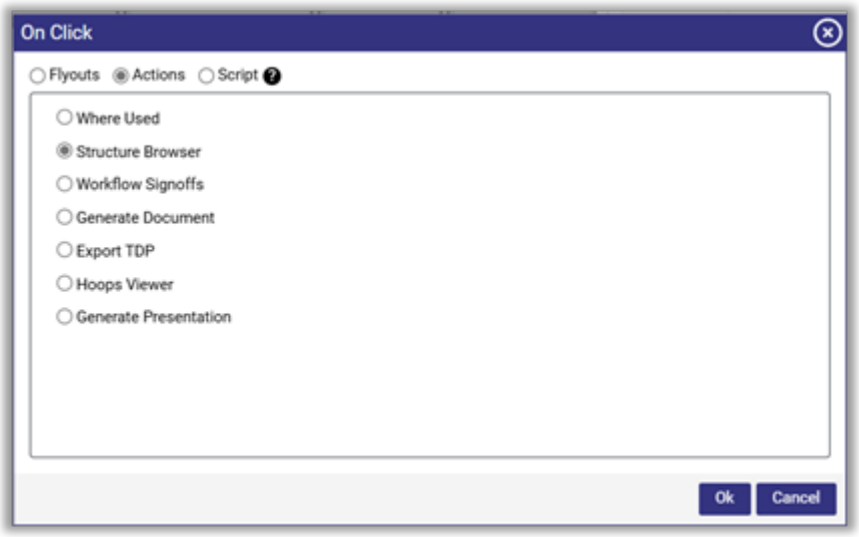
After adding columns, you can edit Width of the column, or you can reorder column using drag-drop icon from the Actions columns. You can hide or show column by selecting the checkbox under Hide column. You can delete existing column by selecting delete icon from the Actions column. By setting Freeze Column option, you can fix columns that are always visible and don't move along with horizontal scrollbar. Freeze Column option is shown only in the Column Width Pixels mode.

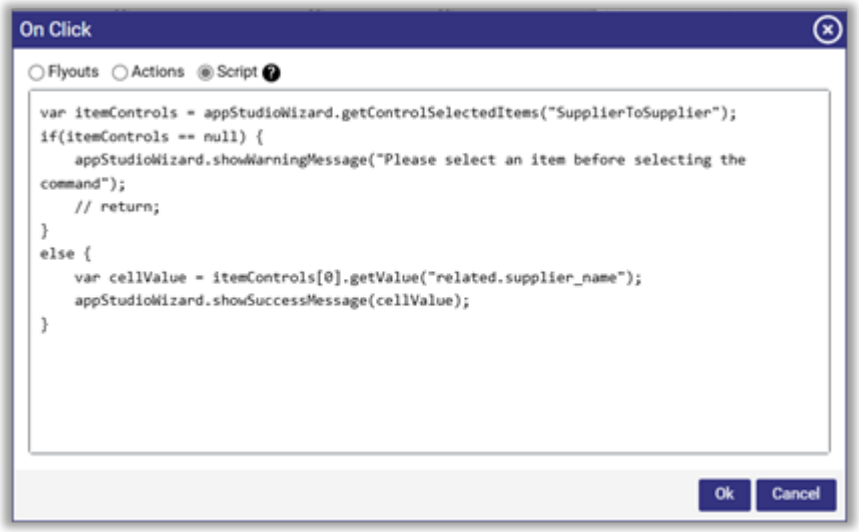
- **Insert Filter**: It allows you to define the filter criteria for Insert Flyout. As an admin, you can define filter criteria that is used when the Insert Flyout is opened. As an admin, you can also enforce the filter, in that case end-user can see the applied filter on the Insert Flyout, but can't change the filter criteria. If enforce filter is not set, end-user can change the filter criteria at the runtime.
- **Insert Columns**: It allows you to define the columns that need to be shown on the Insert Flyout. You can also define the width of the Flyout in percentage with respect to main page.

- **Commands**: Using Commands section, you can show or hide standard commands shown on the Table toolbar. You can also define custom command by selecting Type as Button or DropDown and entering command name. For each custom command, you can set On Click, Can Show, On Load options.
  Each custom command can be configured with 'On Click' handler to perform certain action. It can show a shortcut to the existing Innovator actions like Structure Browser, Where Used, Workflow Signoffs etc. It can also be configured to call a custom script. By selecting Script option in the dialog, you can associate custom script that contains code to interact with other Table columns or controls in the Wizard. Script can also have code to interact with server by sending a request.
  Each custom command can be configured with 'Can Show' option to define custom JavaScript that will return a boolean value. If returned value is true, command will be shown otherwise command will be hidden.
  Each custom command of type 'DropDown' can be configured with 'On Load' option to load options for the DropDown at runtime.
  For each command, you can configure icon. You can also show or hide the command by selecting checkbox under Hide column.









- **Cascade Delete**: It facilitates all child relationships and related items to be deleted when the parent item is deleted.
  - **Cascade Delete**: Yes
    If 'Cascade Delete' is set to Yes, when the context item is deleted, all the child items in the table will get deleted. In order to edit any row in the table, you don't have to lock the row, if the context item is in the edit mode, all child items can be edited without applying the lock.
  - **Cascade Delete**: No
    If 'Cascade Delete' is set to No, for editing any child item, first you have to apply the lock. Also, deleting the context item won't delete child items in the table.
- **Scroll Mode**: Table control in Aras ProAppDesigner can be shown in two distinct modes, with pagination or scroll.
  - **Scroll Mode**: Yes
    If 'Scroll Mode' is set to Yes, table will always be shown with scroll to facilitate row reordering with drag-drop.
  - **Scroll Mode**: No
    If 'Scroll Mode' is set to No, table will always be shown with pagination.

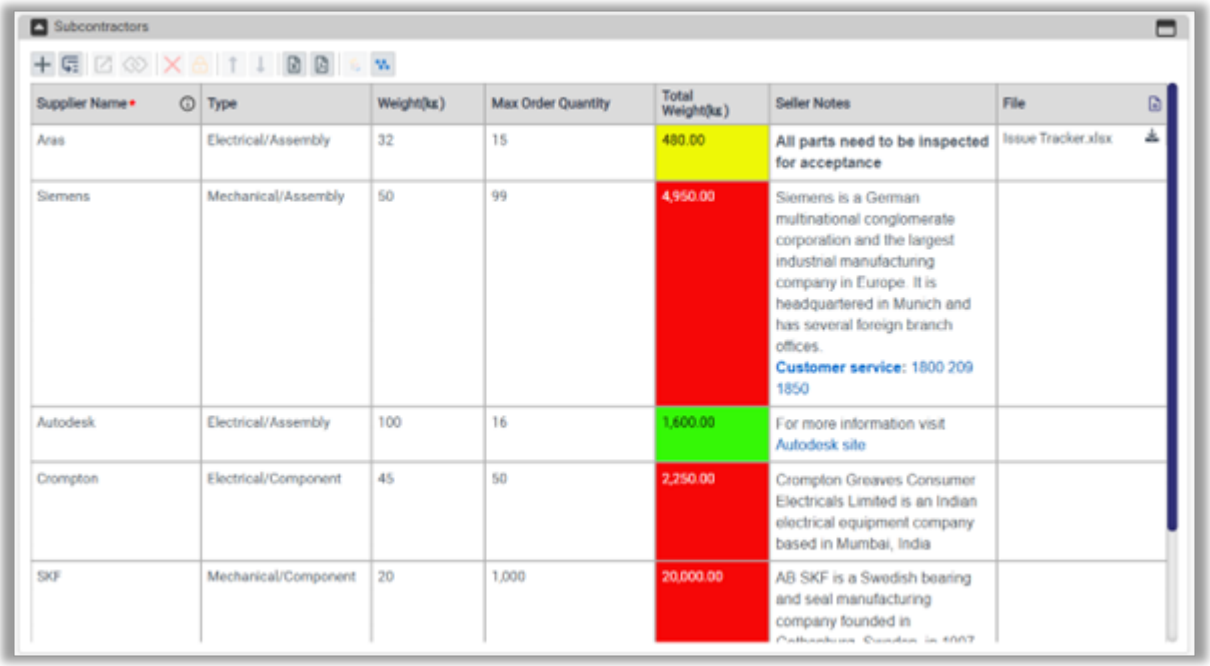
- **Height**: Table control can be rendered in three different mode for the Height.
  - **Height**: Explicit
    In this mode, 'Max Height' should be supplied to limit the height of the table in the page. Table auto grows until it reaches 'Max Height', after that it shows scrollbar for the body of the table.
  - **Height**: Fit to Content
    In this mode, table control grows its height without showing any scrollbar. If it doesn't get required height by the page, then page will start showing scrollbar.
  - **Height**: Automatic
    In this mode, table controls auto grows until it reaches available page height, after that it shows scrollbar for the body of the table. This mode can be set on the table control only when the page that contains table control has only simple form fields apart from this table control.
- **Related Option**: It controls how new rows can be added to the table. This option is defined on the RelationshipType inside innovator, and it is read-only.
  There are three options available:
  - Pick only: You can select the Item which is already present.
  - Create only: You needs to create the Item.
  - Pick and Create: You can either select the existing Item or create a new Item.
- **Show Label**: It controls whether Label can be shown at the top of the table control.
- **Access**: It allows you to define roles which can access control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because user doesn't have defined role, relationship data associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like 'Yes', 'No', 'Expression', or 'Script'. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like 'Always Show', 'Always Hide', 'Expression', or 'Script'. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **On Before Load**: OnBeforeLoad event allows you to write custom logic that will be executed before the data in the control is loaded. This will be useful to set the filter expression before loading the data. Please refer to the Template Customization section for more details.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the data in the control is loaded. Please refer to the Template Customization section for more details.
- **On Insert**: OnInsert event will be launched after item has been inserted in the Table control. Event handler will receive inserted item details via eventData, this object will be available in the scope of the event handler code. Please refer to the Template Customization section for more details.
- **On Insert Filter**: On Insert Filter event will be launched after hitting the search button in Insert Item flyout. Admin can configure a custom script to modify entered filter expression or assign entirely new filter expression string to the eventData. EventData

object is available in the scope of the event handler code. Please refer to the Template Customization section for more details.

- **On Select**: On Select event will be launched after selecting the item in the Table control. Event handler will receive selected item details via eventData, this object will be available in the scope of the event handler code. Please refer to the Template Customization section for more details.

Runtime View:

Table supports selecting single row by clicking on any cell of the row. It also supports multi-row selection with Shift and Ctrl keys. By right clicking on any cell, table control shows context menu with same commands as the toolbar. Table control also supports Column resizing when its columns are configured with Column Width Percentage mode.



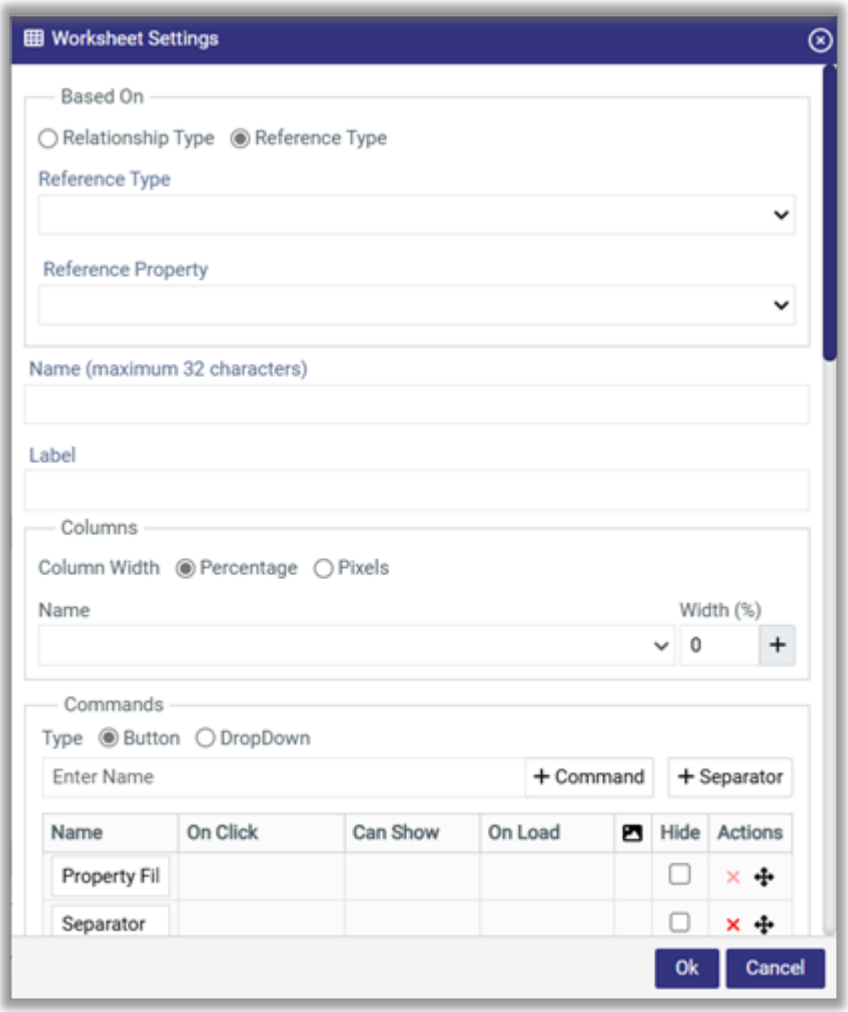
- **Add**: Select Add action to add row after the selected row. Added rows appear in light green color in the table. Updated rows will be shown in light orange color.
- **Insert**: Select Insert action to add row based on the existing item. In order to select existing item, it shows search dialog. Added rows appear in light green color in the table.
- **Open Relationship Item**: Select Open RelationshipItem action to open relationship item in its own tab.
- **Open Related Item**: Select Open Related Item action to open related item in its own tab.
- **Up**: Select Up action to move selected row above the previous row. When ‘Cascade Delete’ setting is ‘No’, Up action will be enabled only when the row is locked.
- **Down**: Select Down action to move selected row below the next row. When ‘Cascade Delete’ setting is ‘No’, Down action will be enabled only when the row is locked.
- **Delete**: Select Delete action to delete selected row from the table. When ‘Cascade Delete’ setting is ‘No’, Delete action will be enabled only when the row is locked.
- **Lock**: Select Lock action to lock selected row for exclusive edit. If the associated item with row is locked by someone else, you won’t be able to lock the row. Lock action will be shown only when ‘Cascade Delete’ setting is ‘No’.
- **Export Excel**: Select ExportExcel action to export all the visible data shown in the table to an excel file.

- **Export PDF**: Select ExportPDF action to export all the visible data shown in the table to a PDF file.
- **Filter**: By clicking on the Hidden dropdown, you can select Simple option in order to show column filters on the table. You can enter filter values on individual columns and select search icon to filter the rows based on the criteria.
- **Pagination**: Pagination controls are shown at the bottom of the table in order to navigate to other pages of the table. You can enter Page Size and select search icon from the top action bar, in order to increase or reduce number of rows per page. Pagination controls are shown only when ‘Cascade Delete’ setting is ‘No’.

Worksheet Control

Worksheet control provides Excel experience for creating datasheet. Worksheet control can be used to show relationship data or reference data of the context item as datasheet. Columns in worksheet will be constrained with property data types. It supports nested rows from child relationships.

When ReferenceType option is selected for creating Worksheet control, a dropdown will be shown with all ItemTypes on which context ItemType is defined as item property. If there are multiple item properties of the context ItemType on the selected ItemType, a dropdown will be shown to select the specific item property.

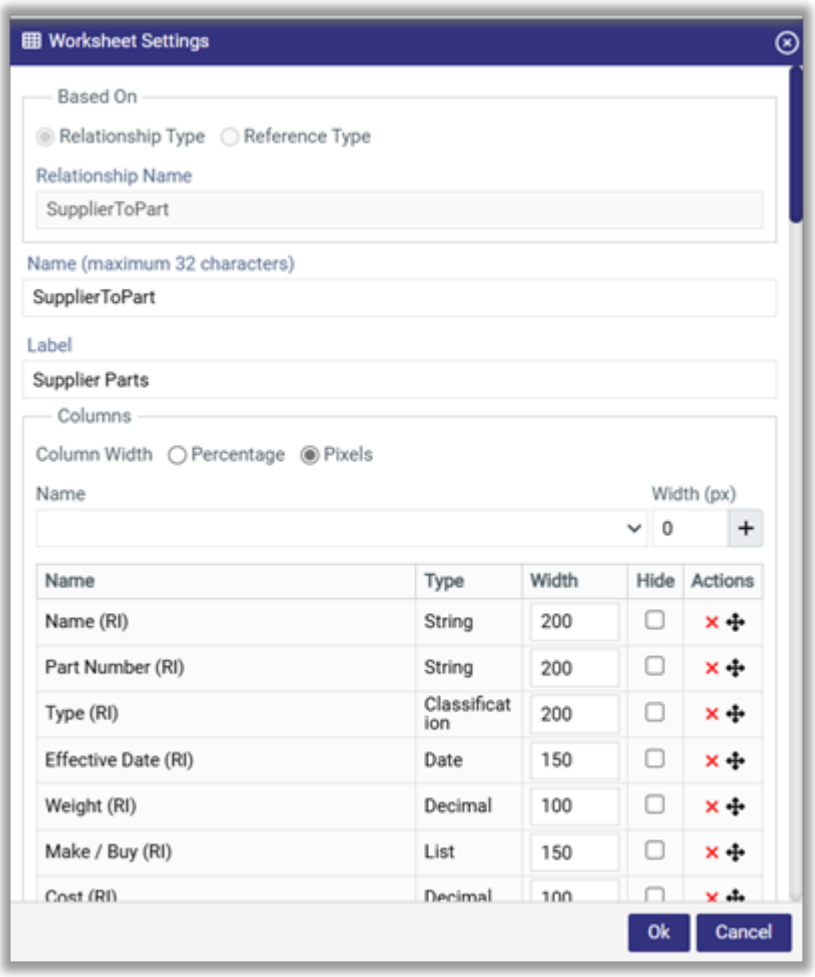


You can navigate cells by navigation keys. It supports Excel type of Sorting and Filtering. You can copy paste between Excel file and worksheet control and vice versa. You can reorder columns by drag & drop operation. You can do all cell copy by dragging.

As part of adding worksheet control to the page, Aras ProAppDesigner shows Settings flyout where you can configure the RelationshipType that you want to associate with the worksheet control. RelationshipType can only be selected at the time of adding worksheet control. If you need to change associated RelationshipType, then you need to delete worksheet control and re-add it. Worksheet control has multiple settings that controls its appearance and behavior at runtime.

You can also configure individual columns on the worksheet control by selecting gear icon from the column. You can set Validation Rules, Default Value, Conditional Formatting, Read Only on the individual column by selecting gear icon from the column that opens Settings flyout. While defining expressions on the individual columns, you will only see properties from other columns of the same worksheet control.

Worksheet will always be shown with scroll to facilitate row reordering with drag-drop. In order to edit any row in the worksheet, you do not have to lock the row, if the context item is in the edit mode, all child items can be edited without applying the lock.

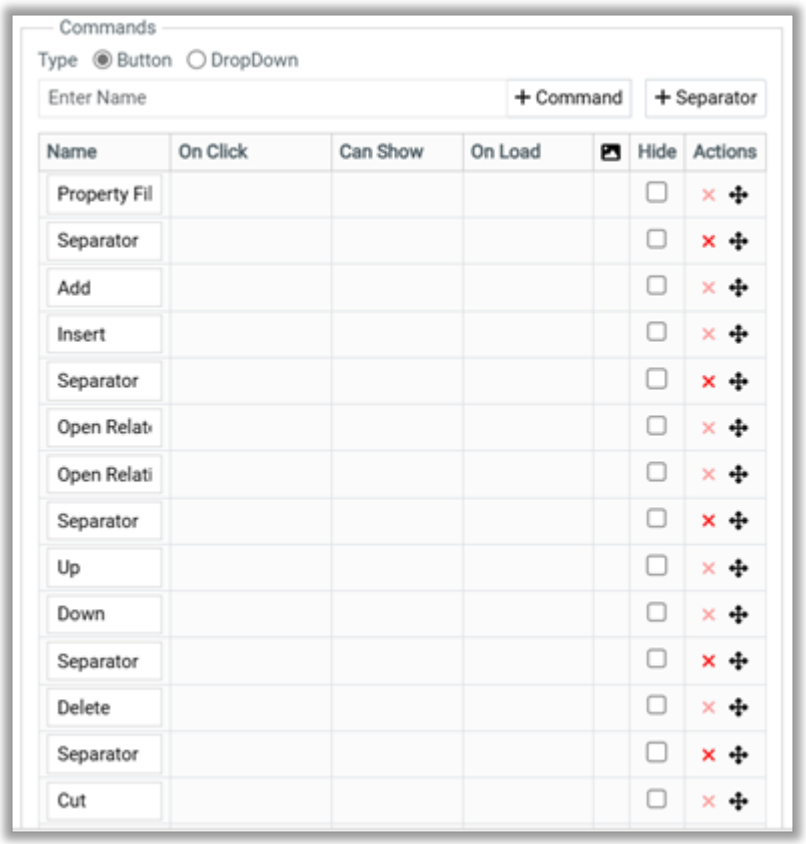


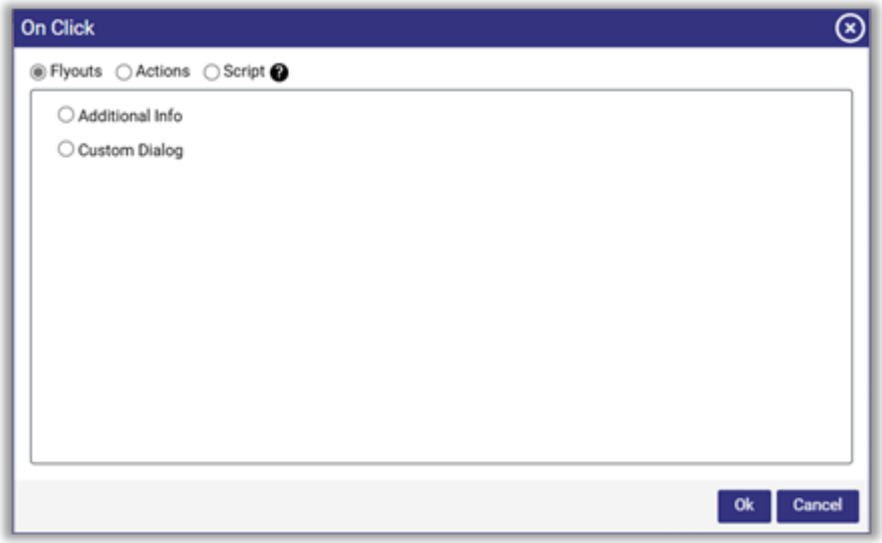
- **Name**: It is used to uniquely to identify control inside the template.
- **Label**: It is used to show title on the worksheet. It can be localized by defined Languages and Locales inside innovator.
- **Columns**: Using Columns settings section, you can add all the required columns to the worksheet. Columns can be added based on the RelationshipType or RelatedItemType properties. Using Column Width option, you can set column widths in percentage or pixels. If you select Column Width as Percentage, you never see horizontal scrollbar, widths of the columns will get adjusted to the available total width of the worksheet. If Column Width option is selected as Pixels, you will see horizontal scrollbar if the combined widths of all columns is more than available worksheet width. In pixel mode, columns can be reordered and resized at runtime. As part of adding column, you need to select the property from either RelationshipType or RelatedItemType, to which you want to bind the column. Columns added with RelatedItemType properties will be shown with (RI) in the column name. After adding columns, you can edit Width of the column, or you can reorder column using drop-drop icon from the Actions columns. You can hide or show column by selecting the checkbox under Hide column. You can delete existing column by selecting delete icon from the Actions column. By setting Freeze Column option, you can fix columns that are always visible and don't move along with horizontal scrollbar. Freeze Column option is shown only in the Column Width Pixels mode.
- **Insert Filter**: It allows you to define the filter criteria for Insert Flyout. As an admin, you can define filter criteria that is used when the Insert Flyout is opened. As an admin, you can also enforce the filter, in that case end-user can see the applied filter

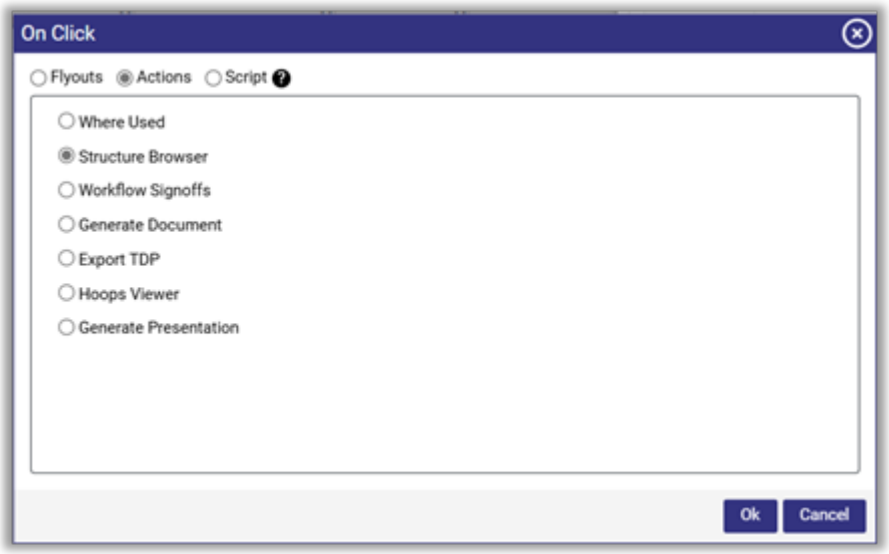
on the Insert Flyout, but can't change the filter criteria. If enforce filter is not set, end-user can change the filter criteria at the runtime.

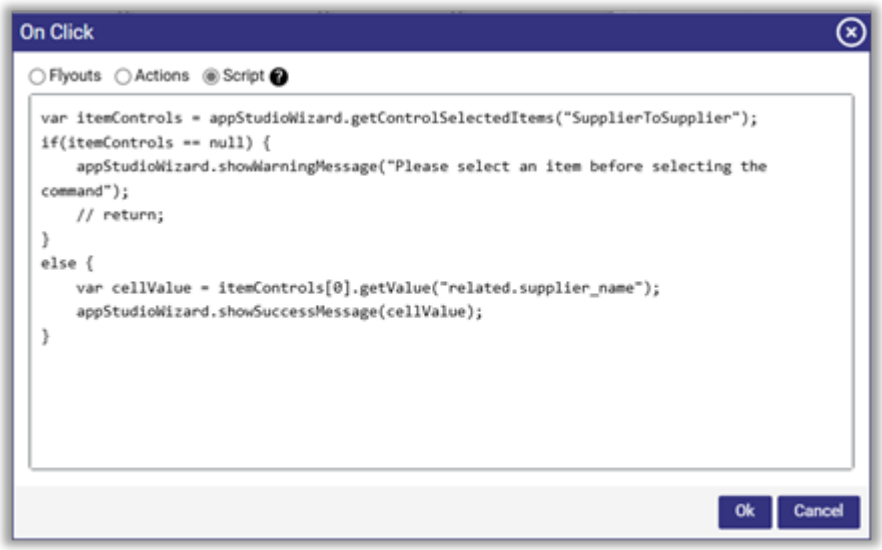
- **Insert Columns**: It allows you to define the columns that need to be shown on the Insert Flyout. You can also define the width of the Flyout in percentage with respect to main page.

- **Commands**: Using Commands section, you can show or hide standard commands shown on the Worksheet toolbar. You can also define custom command by selecting Type as Button or DropDown and entering command name. For each custom command, you can set On Click, Can Show, On Load options.
  Each custom command can be configured with 'On Click' handler to perform certain action. It can show a shortcut to the existing Innovator actions like Structure Browser, Where Used, Workflow Signoffs etc. It can also be configured to call a custom script. By selecting Script option in the dialog, you can associate custom script that contains code to interact with other Worksheet columns or controls in the Wizard. Script can also have code to interact with server by sending a request.
  Each custom command can be configured with 'Can Show' option to define custom JavaScript that will return a boolean value. If returned value is true, command will be shown otherwise command will be hidden.
  Each custom command of type 'DropDown' can be configured with 'On Load' option to load options for the DropDown at runtime.
  For each command, you can configure the icon. You can also show or hide the command by selecting checkbox under Hide column.



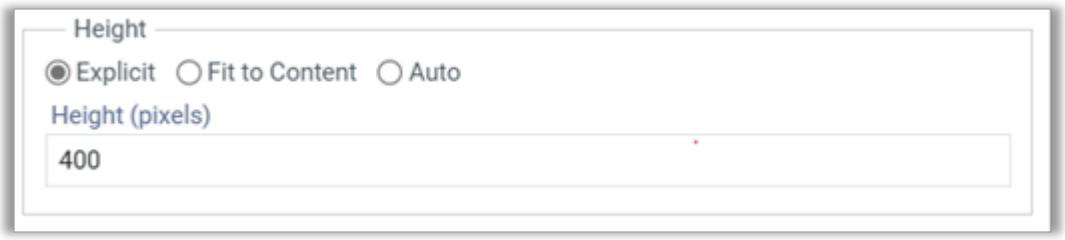






- **Cascade Delete**: It defines how the rows in the worksheet should be handled when the context item is deleted.
  - **Cascade Delete**: Yes
    If Cascade Delete is set to Yes, when the context item is deleted, all the child items in the worksheet will get deleted.
  - **Cascade Delete**: No
    If Cascade Delete is set to 'No', deleting the context item won't delete child items in the worksheet.

- **Height**: Worksheet control can be rendered in three different mode for the Height.



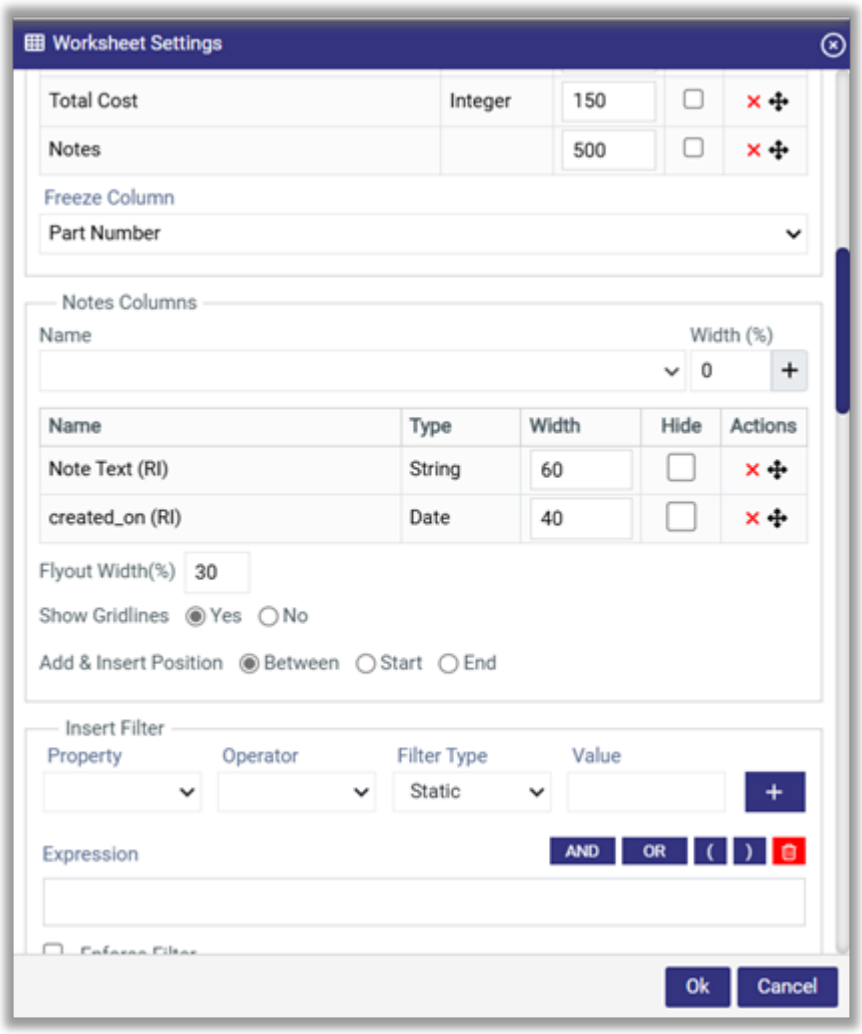
  - **Height**: Explicit
    In this mode, ‘Height’ should be supplied to limit the height of the worksheet in the page. Worksheet is rendered with the given height upfront, if it grows beyond this height then it will start showing scrollbar for the body of the worksheet.
  - **Height**: Fit to Content
    In this mode, worksheet control grows its height without showing any scrollbar. If it does not get required height by the page, then page will start showing scrollbar.
  - **Height**: Automatic
    In this mode, worksheet controls auto grows until it reaches available page height, after that it shows scrollbar for the body of the worksheet. This mode can be set on the worksheet control only when the page that contains worksheet control has only simple form fields apart from this worksheet control.
- **Related Option**: It controls how new rows can be added to the worksheet. This option is defined on the RelationshipType inside innovator, and it is read-only.
  There are three options available:
  - **Pick only**: You can select the Item which is already Present.
  - **Create only**: You need to create the Item.
  - **Pick and Create**: You can either select the existing Item or create a new Item.
- **Show Label**: It controls whether Label can be shown at the top of the worksheet control.
- **Access**: It allows you to define roles which can access the control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, relationship data associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Font**: It allows you to set the required font and size for the cells including column headings of the worksheet control.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **On Before Load**: OnBeforeLoad event allows you to write custom logic that will be executed before the data in the control is loaded. This will be useful to set the filter expression before loading the data. Please refer to the Template Customization

section for more details.

- **On Load**: OnLoad event allows you to write custom logic that will be executed after the data in the control is loaded. Please refer to the Template Customization section for more details.
- **On Insert**: On Insert event will be launched after Item has been inserted in the Worksheet control. Event handler will receive inserted item details via eventData, this object will be available in the scope of the event handler code. Please refer to the Template Customization section for more details.
- **On Insert Filter**: On Insert Filter event will be launched after hitting the search button in Insert Item flyout. Admin can configure a custom script to modify entered filter expression or assign entirely new filter expression string to the eventData. EventData object is available in the scope of the event handler code. Please refer to the Template Customization section for more details.
- **On Select**: On Select event will be launched after selecting the item in the Worksheet control. Event handler will receive selected item details via eventData, this object will be available in the scope of the event handler code. Please refer to the Template Customization section for more details.

Nested Rows:

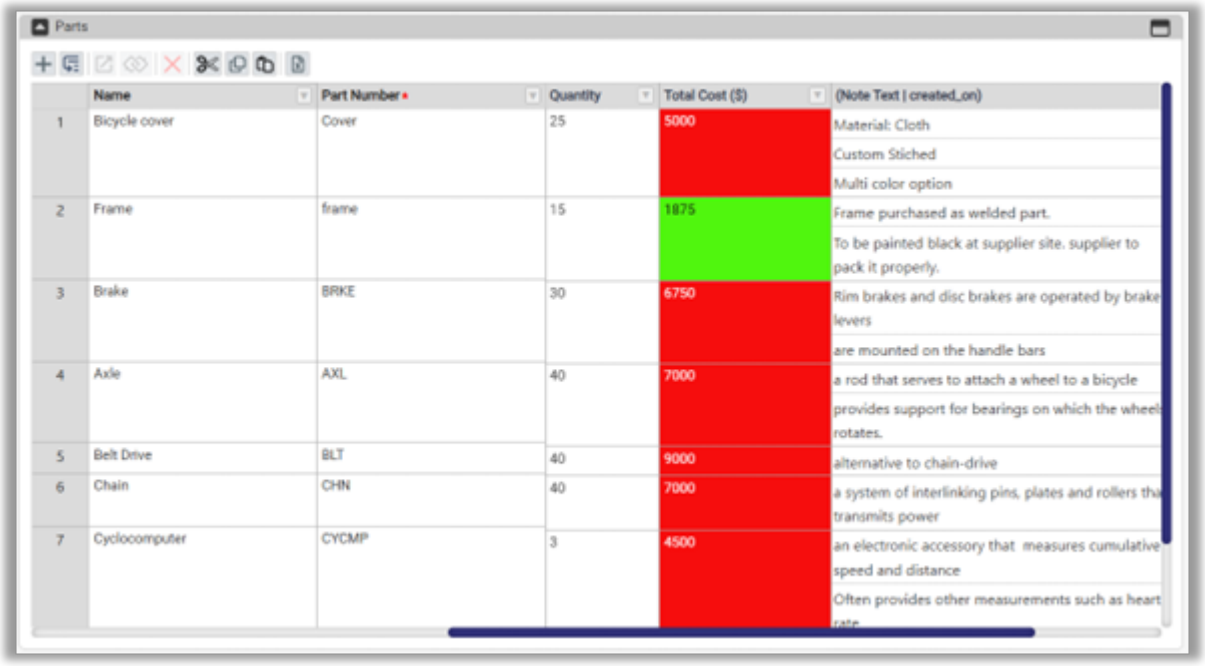
Worksheet control can be configured to show child relationship data as nested rows of the main worksheet row. In order to configure nested rows, you are supposed to select child relationship type as column for the main worksheet. Then Aras ProAppDesigner adds child relationship columns section where you can configure which properties should be shown as columns for nested rows. At runtime, nested rows data can be edited from the flyout which is shown by double clicking nested row from the main worksheet. You can set width in percentage for the flyout with respect to the base page by setting Flyout Width value. By selecting Show Gridlines option, you can show nested rows in the base worksheet as columns and rows otherwise they will be shown with ‘,’ and newline delimiters. If you want to enforce adding new items at the specific position in the Nested Rows flyout, you can select Add & Insert Position setting.



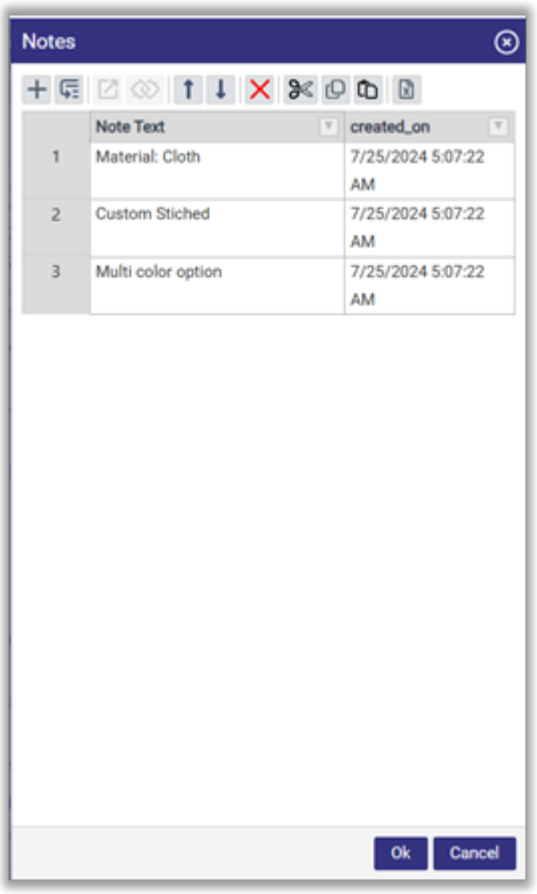
Runtime View:

Worksheet supports selecting single row by clicking on any cell of the row. It also supports multi-row selection with Shift and Ctrl keys. By right clicking on any cell, worksheet control shows context menu with same commands as the toolbar. Worksheet control also supports Column resizing when its columns are configured with Column Width Percentage mode.

Main Worksheet:



Nested Rows Worksheet:

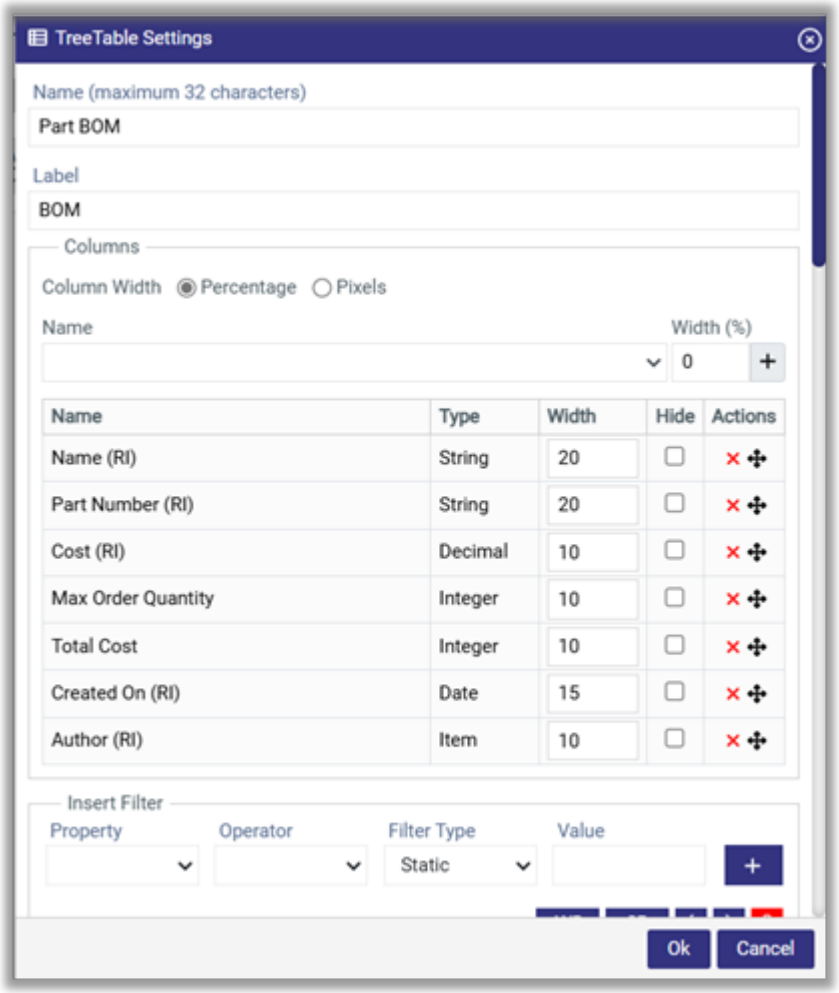


- **Add**: Select Add action to add row after the selected row.
- **Insert**: Select Insert action to add row based on the existing item. In order to select existing item, it shows search dialog.
- **Open Relationship Item**: Select OpenRelationshipItem action to open relationship item in its own tab.
- **Open Related Item**: Select Open Related Item action to open related item in its own tab.
- **Up**: Select Up action to move selected row above the previous row.
- **Down**: Select Down action to move selected row below the next row.
- **Delete**: Select Delete action to delete selected row from the worksheet.
- **Cut**: Select Cut action to cut selected cell or row from the worksheet.
- **Copy**: Select Copy action to copy selected cell or row from the worksheet.
- **Paste**: Select Paste action to paste data from the clipboard to the selected cell or insert row after the selected row.
- **Export**: Select Export action to export all the data shown in the worksheet to an excel file.

TreeTable Control

TreeTable control is used to show the structure of items of the same ItemType. As part of adding tree-table control to the form, Aras ProAppDesigner shows Settings flyout where you can configure the RelationshipType that you want to associate with the tree-table control. In the RelationshipType dropdown, you can only see those RelationshipTypes with same ItemType on both sides. RelationshipType can only be selected at the time of adding tree-table control. If you need to change associated RelationshipType, then you need to delete tree-table control and re-add it. TreeTable control has multiple settings that controls its appearance and behavior at runtime.

You can also configure individual columns on the tree-table control by selecting gear icon from the column. You can set Validation Rules, Default Value, Conditional Formatting, Read Only on the individual column by selecting gear icon from the column that opens Settings flyout. While defining expressions on the individual columns, you will only see properties from other columns of the same tree-table control.

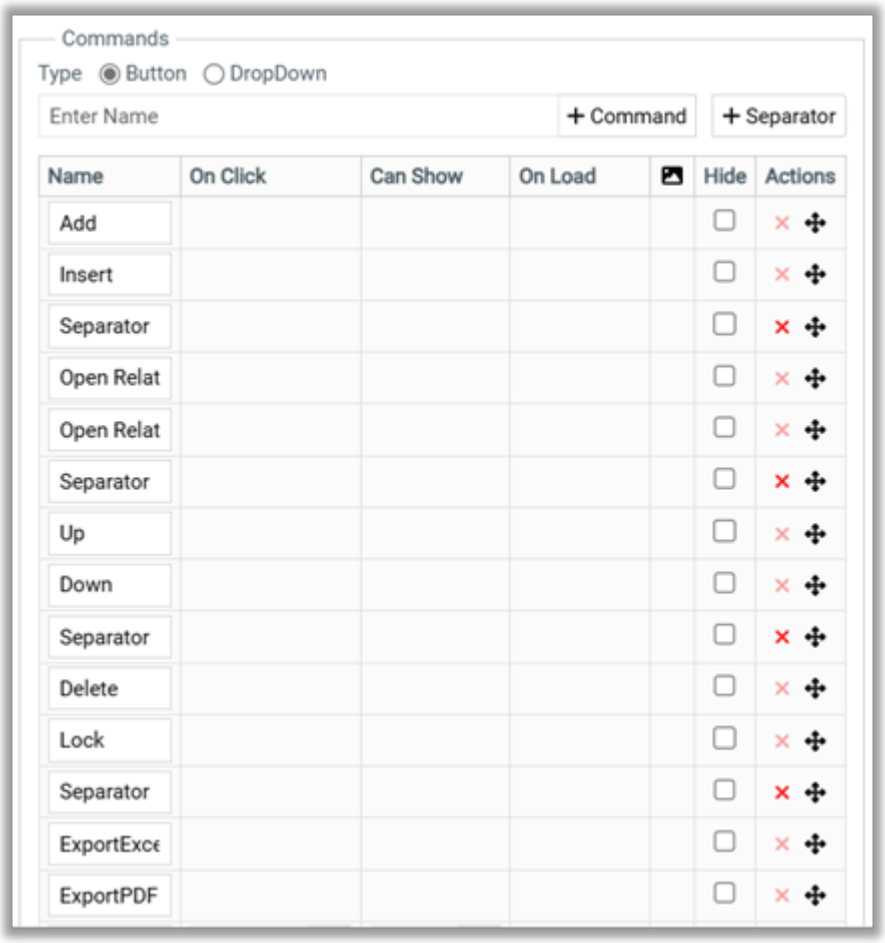


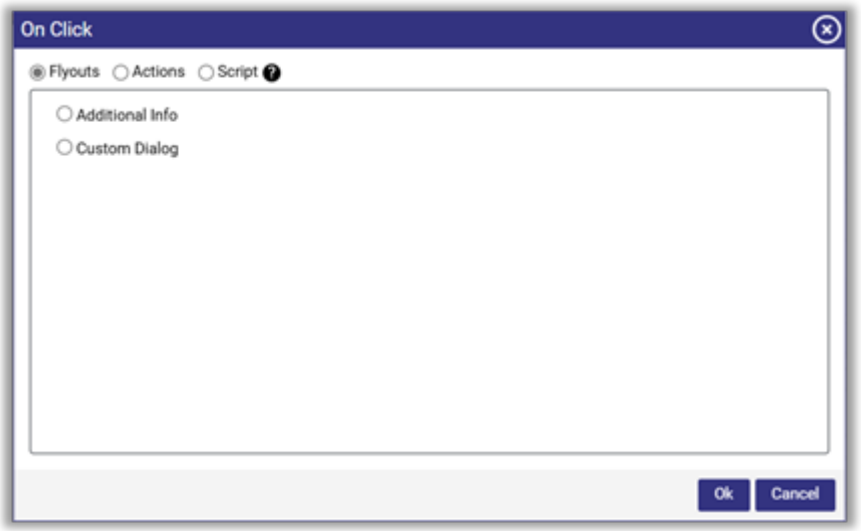
- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the tree-table. It can be localized by defined Languages and Locales inside innovator.
- **Columns**: Using Columns settings section, you can add all the required columns to the tree-table. Columns can be added based on the RelationshipType or RelatedItemType properties. Using Column Width option, you can set column widths in percentage or pixels. If you select Column Width as Percentage, you never see horizontal scrollbar, widths of the columns will get adjusted to the available total width of the tree-table. If Column Width option is selected as Pixels, you will see horizontal scrollbar if the combined widths of all columns is more than available tree-table width. As part of adding column, you need to select the property from either RelationshipType or RelatedItemType, to which you want to bind the column. Columns added with RelatedItemType properties will be shown with (RI) in the column name. After adding columns, you can edit Width of the column, or you can reorder column using drag-drop icon from the Actions columns. You can hide or show column by selecting the checkbox under Hide column. You can delete existing column by selecting delete icon from the Actions column. By setting Freeze Column option, you can fix columns that are always visible and don't move along with horizontal scrollbar. Freeze Column option is shown only in the Column Width Pixels mode.
- **Insert Filter**: It allows you to define the filter criteria for Insert Flyout. As an admin, you can define filter criteria that is used when the Insert Flyout is opened. As an admin, you can also enforce the filter, in that case end-user can see the applied filter

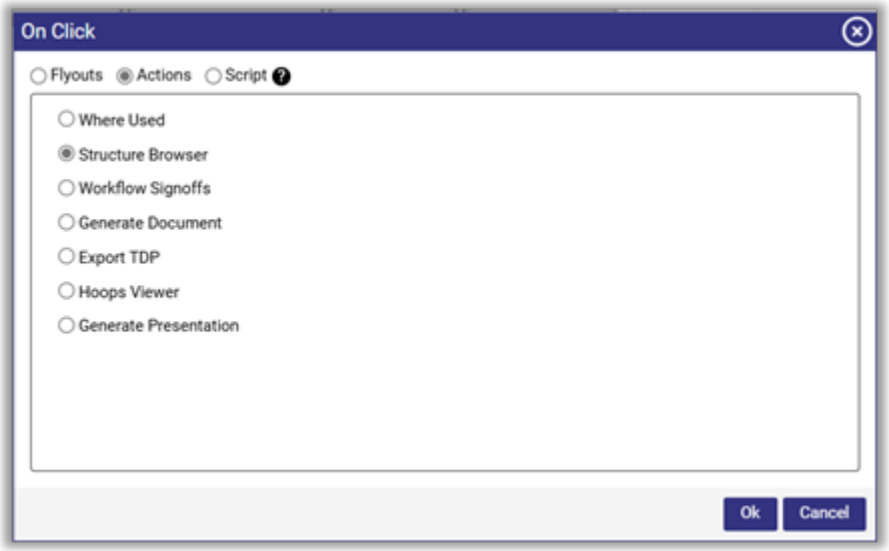
on the Insert Flyout, but can't change the filter criteria. If enforce filter is not set, end-user can change the filter criteria at the runtime.

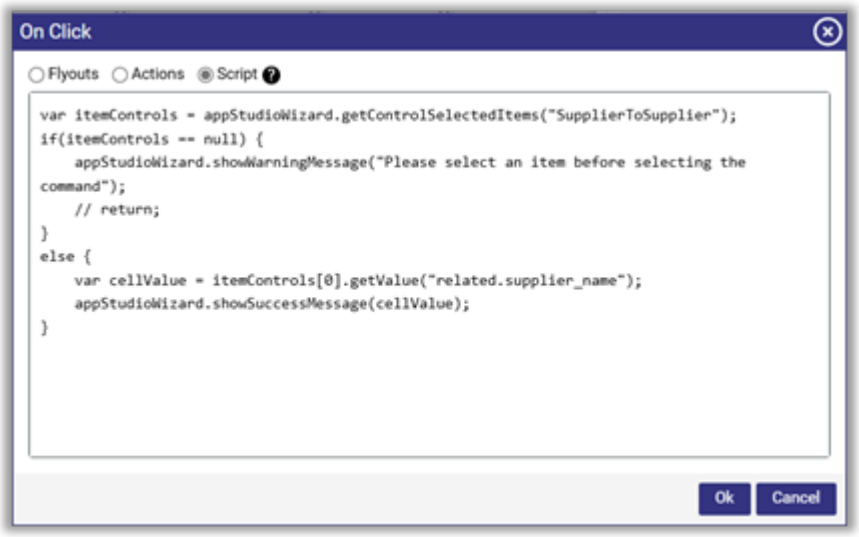
- **Insert Columns**: It allows you to define the columns that need to be shown on the Insert Flyout. You can also define the width of the Flyout in percentage with respect to main page.

- **Commands**: Using Commands section, you can show or hide standard commands shown on the TreeTable toolbar. You can also define custom command by selecting Type as Button or DropDown and entering command name. For each custom command, you can set On Click, Can Show, On Load options.
  Each custom command can be configured with 'On Click' handler to perform certain action. It can show a shortcut to the existing Innovator actions like Structure Browser, Where Used, Workflow Signoffs etc. It can also be configured to call a custom script. By selecting Script option in the dialog, you can associate custom script that contains code to interact with other Worksheet columns or controls in the Wizard. Script can also have code to interact with server by sending a request.
  Each custom command can be configured with 'Can Show' option to define custom JavaScript that will return a boolean value. If returned value is true, command will be shown otherwise command will be hidden.
  Each custom command of type 'DropDown' can be configured with 'On Load' option to load options for the DropDown at runtime.
  For each command, you can configure the icon. You can also show or hide the command by selecting checkbox under Hide column.







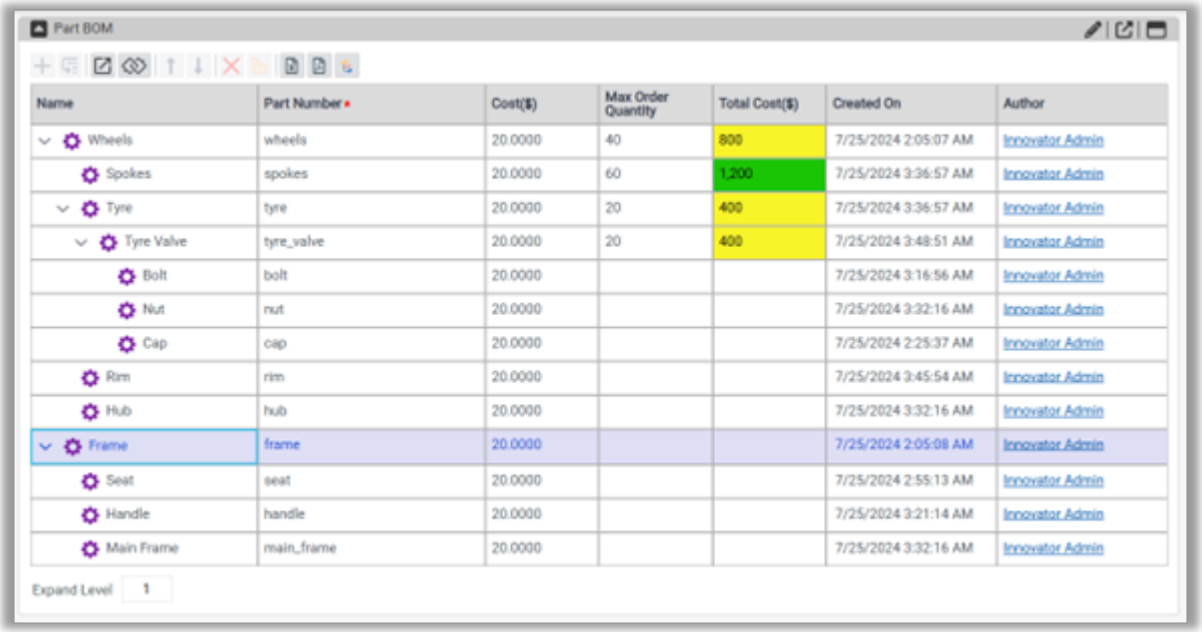


- **Expand Level**: By default, when the tree-table is loaded it only gets its first level children. If you want to expand to multiple levels, expand level value can be set.
- **Height**: TreeTable control can be rendered in three different modes for the Height.
  - **Height**: Explicit
    In this mode, Max Height' should be supplied to limit the height of the tree-table in the page. TreeTable auto grows until it reaches 'Max Height', after that it shows scrollbar for the body of the tree-table.
  - **Height**: Fit to Content
    In this mode, tree-table control grows its height without showing any scrollbar. If it doesn't get required height by the page, then page will start showing scrollbar.
  - **Height**: Automatic
    In this mode, tree-table controls auto grows until it reaches available page height, after that it shows scrollbar for the body of the tree-table. This mode can be set on the tree-table control only when the page that contains tree-table control has only simple form fields apart from this tree-table control.
- **Related Option**: It controls how new rows can be added to the tree-table. This option is defined on the RelationshipType inside innovator, and it is read-only.
  There are three options available:
  - **Pick only**: User can select the Item which is already Present.
  - **Create only**: User needs to create the Item.
  - **Pick and Create**: User can either select the existing Item or create a new Item.
- **Show Label**: It controls whether Label can be shown at the top of the tree-table control.
- **Access**: It allows you to define roles which can access the control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, relationship data associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like 'Yes', 'No', 'Expression', or 'Script'. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.

- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Font**: It allows you to set the required font and size for the cells including column headings of the tree-table control.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **On Before Load**: OnBeforeLoad event allows you to write custom logic that will be executed before the data in the control is loaded. Please refer to the Template Customization section for more details.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the data in the control is loaded. Please refer to the Template Customization section for more details.
- **On Insert**: On Insert event will be launched after Item has been inserted in the TreeTable control. Event handler will receive inserted item details via eventData, this object will be available in the scope of the event handler code. Please refer to the Template Customization section for more details.
- **On Insert Filter**: On Insert Filter event will be launched after hitting the search button in Insert Item flyout. Admin can configure a custom script to modify entered filter expression or assign entirely new filter expression string to the eventData. EventData object is available in the scope of the event handler code. Please refer to the Template Customization section for more details.
- **On Select**: On Select event will be launched after selecting the item in the TreeTable control. Event handler will receive selected item details via eventData, this object will be available in the scope of the event handler code. Please refer to the Template Customization section for more details.

Runtime View:

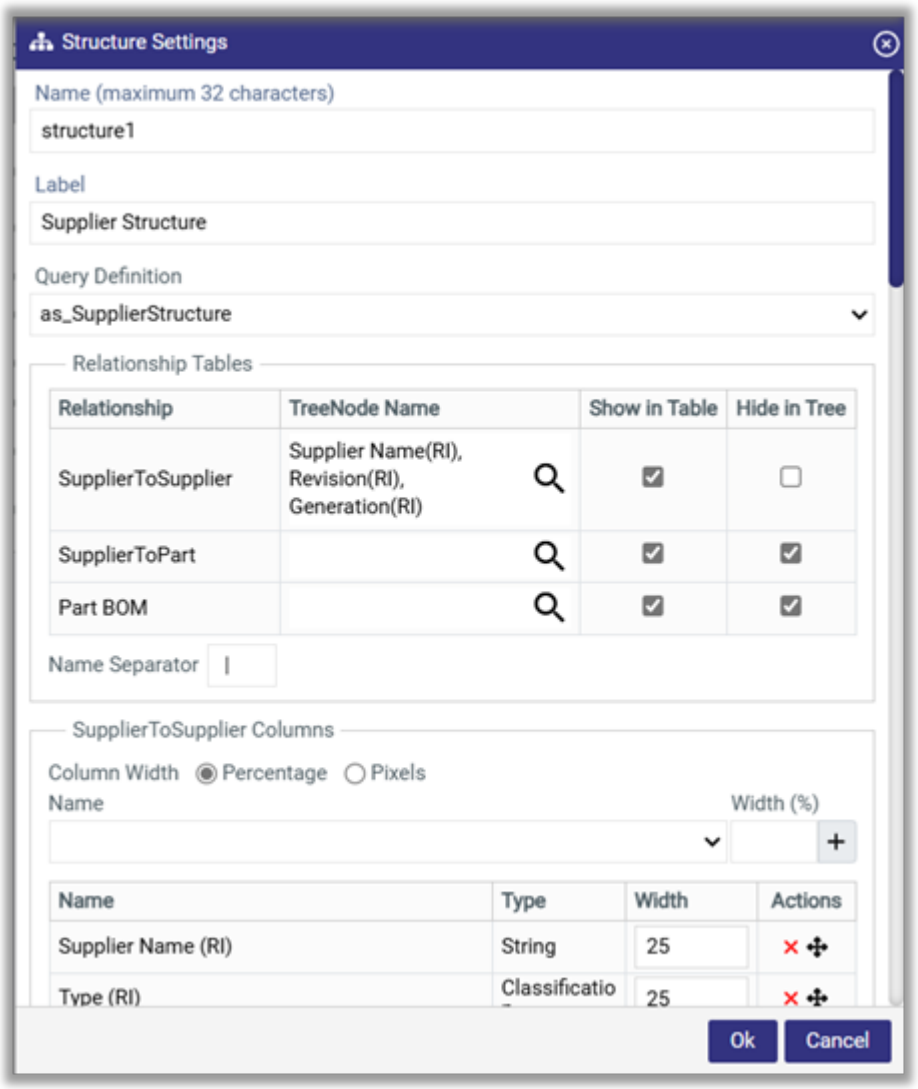
By right clicking on any cell, tree-table control shows context menu with same commands as the toolbar.



- **Add**: Select Add action to add an item as a child or sibling of the selected node.
- **Insert**: Select Insert action to insert an item as a child or sibling of the selected node. In order to select existing item, it shows Insert Flyout.
- **Open Relationship Item**: Select OpenRelationshipItem action to open relationship item in its own tab.
- **Open Related Item**: Select Open Related Item action to open related item in its own tab.
- **Up**: Select Up action to move selected row above the previous row. Up action will be enabled only when the parent item is locked.
- **Down**: Select Down action to move selected row below the next row. Down action will be enabled only when the parent item is locked.
- **Delete**: Select Delete action to delete selected row from the table. Delete action will be enabled only when the item is locked.
- **Lock**: Select Lock action to lock selected row for exclusive edit. If the associated item with the row is locked by someone else, you won't be able to lock the row.
- **Export Excel**: Select ExportExcel action to export all the visible data shown in the table to an excel file.
- **Export PDF**: Select ExportPDF action to export all the visible data shown in the table to a PDF file.
- **Expand Level**: By default, when the node is expanded it only gets its first level children. If you want to expand to multiple levels, expand level value can be set.

Structure Control

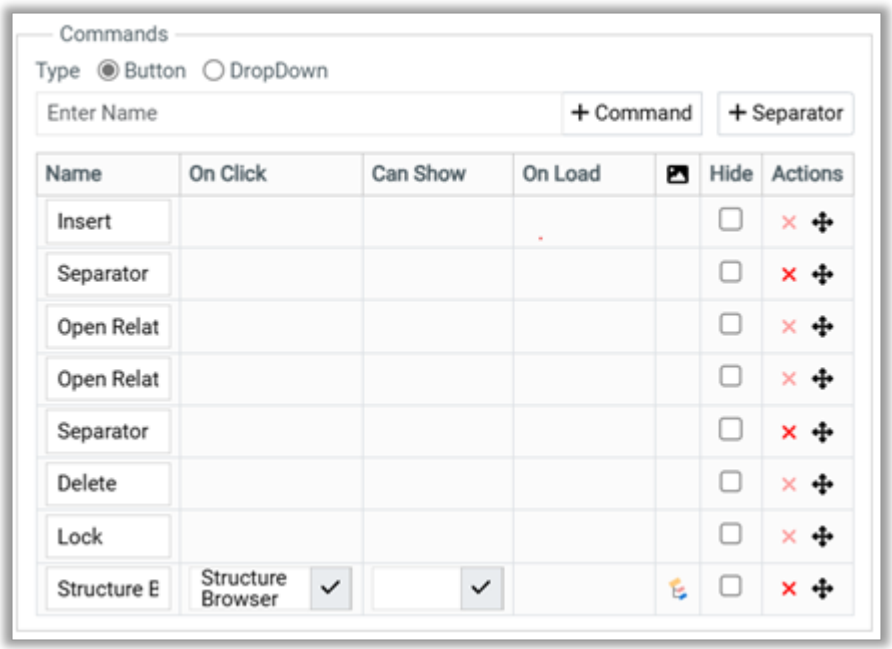
Structure control is used to show structure data of the context item. Structure control is bound to a Query Definition, that defines which RelationshipTypes, RelatedItemTypes and properties to show in the control. Structure control has two panels, on the left side it shows structure of the context item and on right side it shows properties and relationships of the selected node from the tree. Structure control comes with two modes, read-only mode shows structure and properties, you can only navigate the structure and see properties of the selected node. You can also open the specific node in its own tab by selecting Open icon from the toolbar. In the edit mode, you can insert new nodes in the structure by selecting the Insert icon from the toolbar. In the InsertNode Flyout, allowed RelationshipTypes of the selected node are shown, by selecting specific RelationshipType and keyword, you can search and select a specific item to insert under the selected node. In edit mode, you can also edit the properties of the selected node. In order delete a specific node in edit mode, Delete icon can be selected from the toolbar. When you add Structure control on the page, it just shows only those Query Definitions defined with the same ItemType associated with the template. Structure control shows context menu at runtime with same commands as the toolbar.

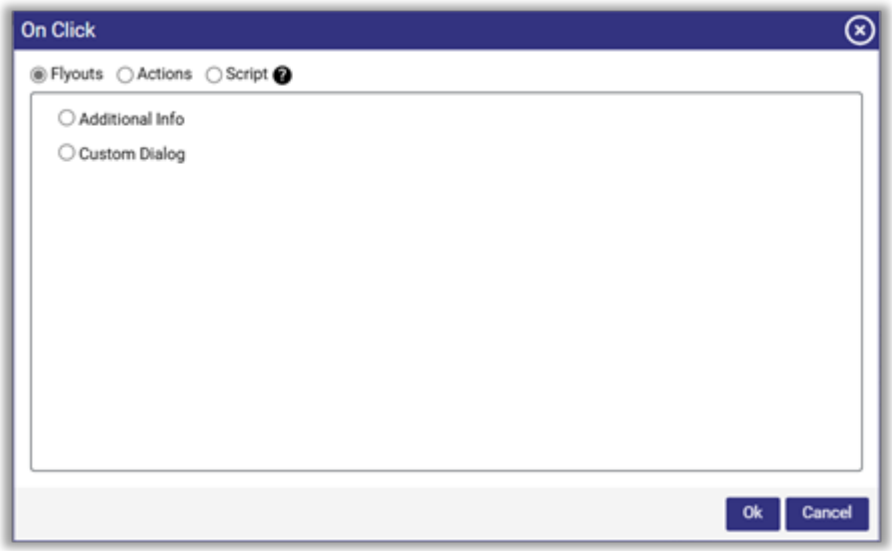


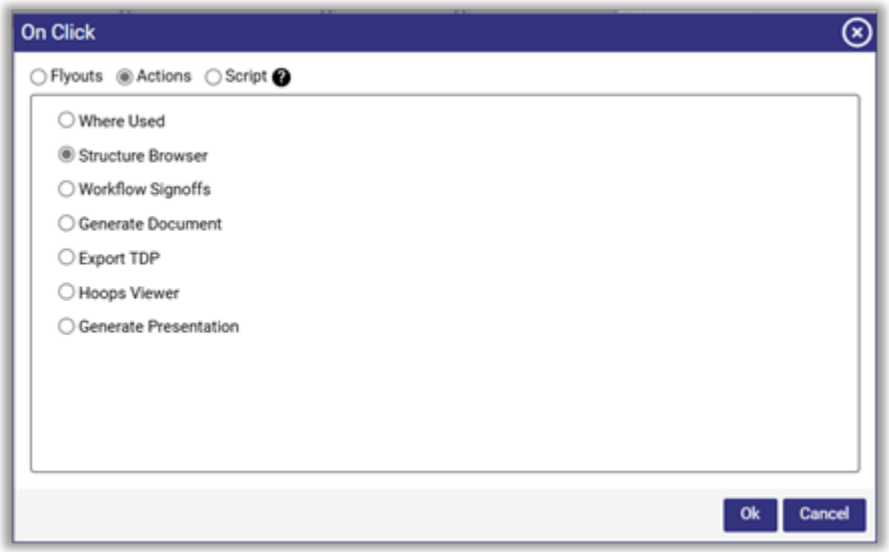
- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the control. It can be localized by defined Languages and Locales inside innovator.
- **Query Definition**: It allows you to select the query that can be used to fetch data from Innovator database. It shows a list of all Query Definitions available in Innovator which are defined for the ItemType associated with the template.
- **Relationship Tables**: It allows you to select which relationships should be shown in the Tables on the right side and/or within the Tree. It also allows you to configure the name shown in the Tree based on multiple properties. For each relationship configured to show in the Table on the right side, it adds a new section to define what columns should be shown in the table.
- **Show Label**: It controls whether Label can be shown at the top of the structure control.

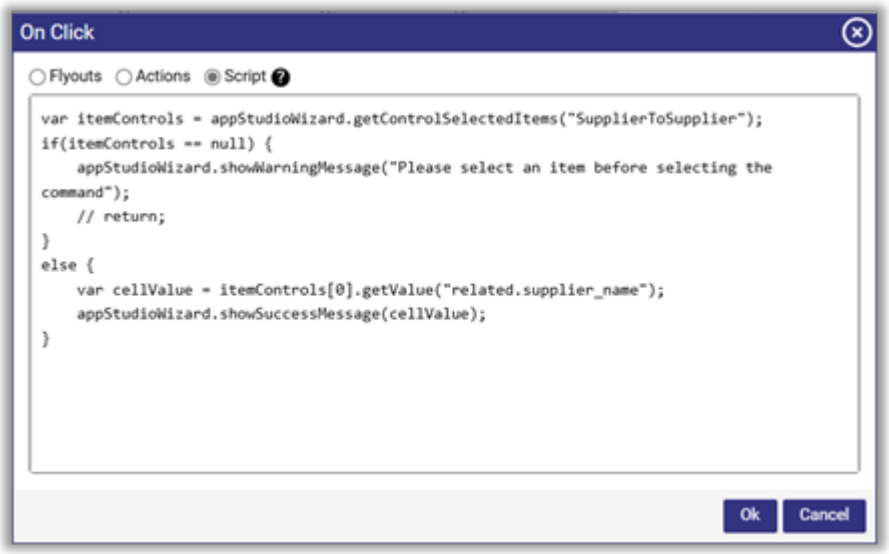
- **Commands**: Using Commands section, you can show or hide standard commands shown on the Structure toolbar. You can also define custom command by selecting Type as Button or DropDown and entering command name. For each custom command, you can set On Click,

Can Show, On Load options.
Each custom command can be configured with ‘On Click’ handler to perform certain action. It can show a shortcut to the existing Innovator actions like Structure Browser, Where Used, Workflow Signoffs etc. It can also be configured to call a custom script. By selecting Script option in the dialog, you can associate custom script that contains code to interact with other Worksheet columns or controls in the Wizard. Script can also have code to interact with server by sending a request.
Each custom command can be configured with ‘Can Show’ option to define custom JavaScript that will return a boolean value. If returned value is true, command will be shown otherwise command will be hidden.
Each custom command of type ‘DropDown’ can be configured with ‘On Load’ option to load options for the DropDown at runtime.
For each command, you can configure the icon. You can also show or hide the command by selecting checkbox under Hide column.





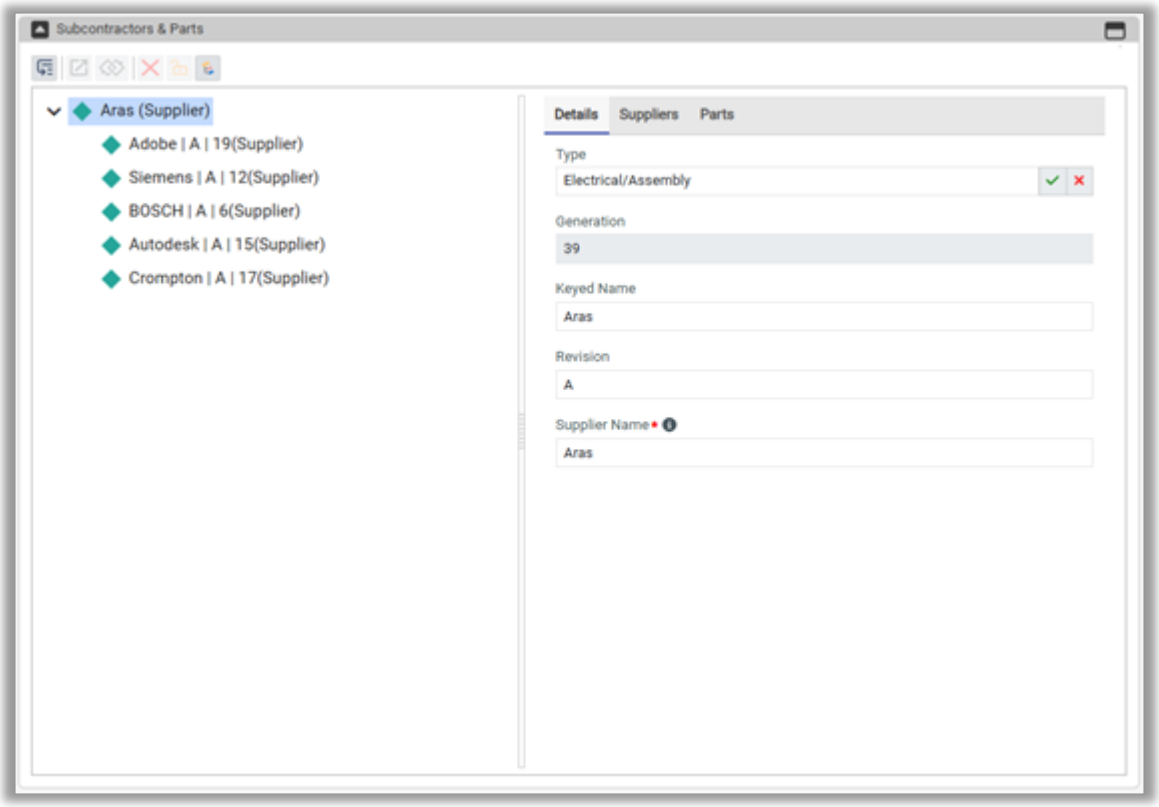




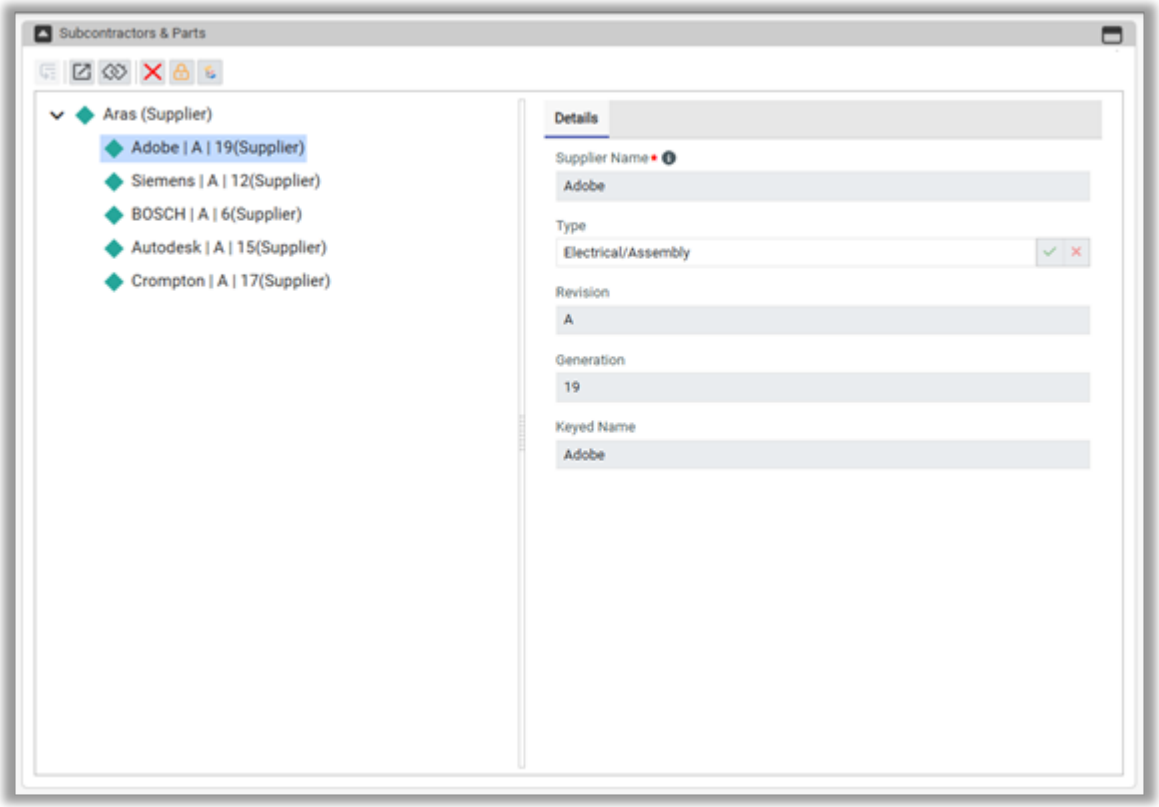
- **Height**: Structure control can be rendered in two different modes for the Height.
  - **Height**: Explicit
    In this mode, Max Height' should be supplied to limit the height of the control in the page. Control auto grows until it reaches 'Max Height', after that it shows scrollbar for the body of the Tree.
  - **Height**: Automatic
    In this mode, control auto grows until it reaches available page height, after that it shows scrollbar for the body of the Tree. This mode can be set on the structure control only when the page that contains structure control has only simple form fields apart from this structure control.
- **Width**: It allows you to set the width in percentage for the Tree portion of the structure from the total available width.
- **Insert Flyout**: It allows you to set the flyout width in percentage. It also allows you to show Add command in the toolbar in the insert flyout. Using this command, new item can be created on the fly as part of the insert operation.
- **Access**: It allows you to define roles which can access control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, data associated with this control is not even fetched from server.

- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like 'Yes', 'No', 'Expression', or 'Script'. If Read Only is selected as 'Yes', you can only navigate structure from the tree panel, and when the node is selected its properties are shown on the right panel. If Read Only is selected as 'No', along with the tree navigation, you can also edit the structure as well as properties of the selected node. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true,

the control will be read-only. In order to see structure control in edit mode, you need to put context item in edit mode.
Read Only: Yes



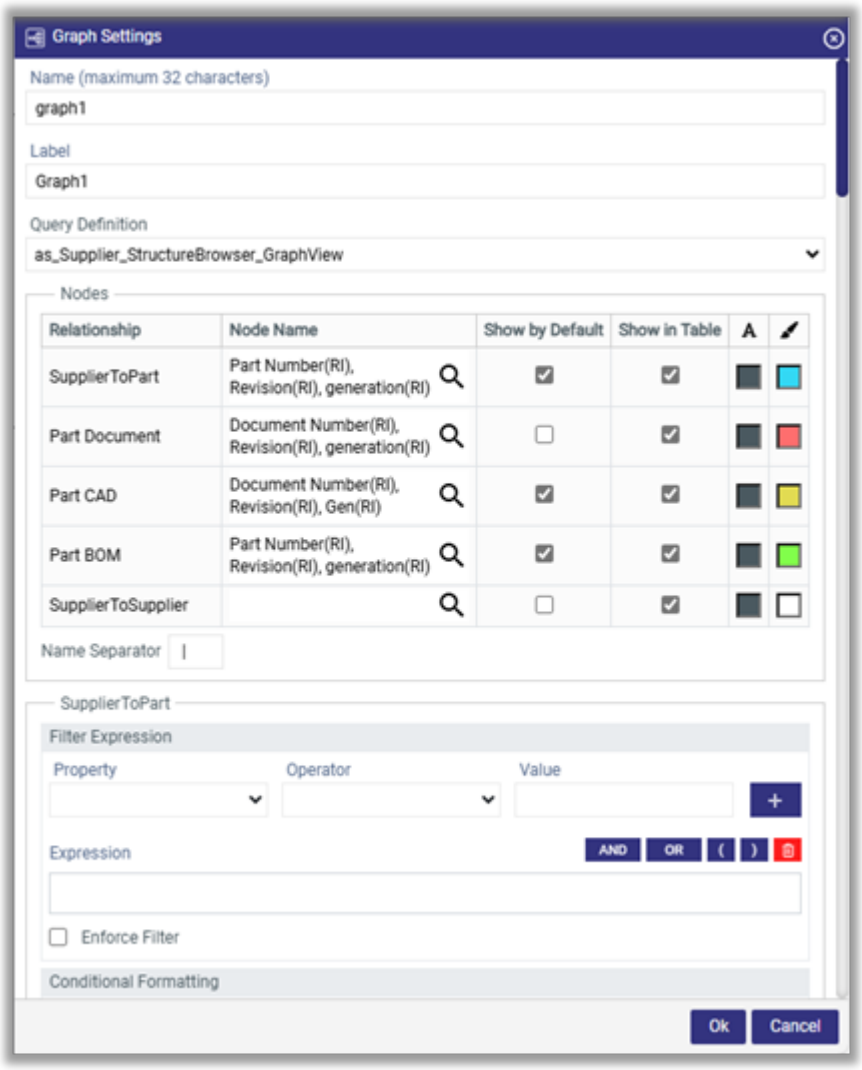
Read Only: No



- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Font**: It allows you to set the required font and size for the node names and form field values of the structure control.
- **Margin**: It allows you to the set required top, right, bottom, left margins with respect to adjacent controls.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the data in the control is loaded. Please refer to the Template Customization section for more details.
- **On Insert**: On Insert event will be launched after Item has been inserted in the Structure control. Event handler will receive inserted item details via eventData, this object will be available in the scope of the event handler code. Please refer to Template Customization section for more details.
- **On Select**: On Select event will be launched after selecting the item in the Structure control. Event handler will receive selected item details via eventData, this object will be available in the scope of the event handler code. Please refer to the Template Customization section for more details.

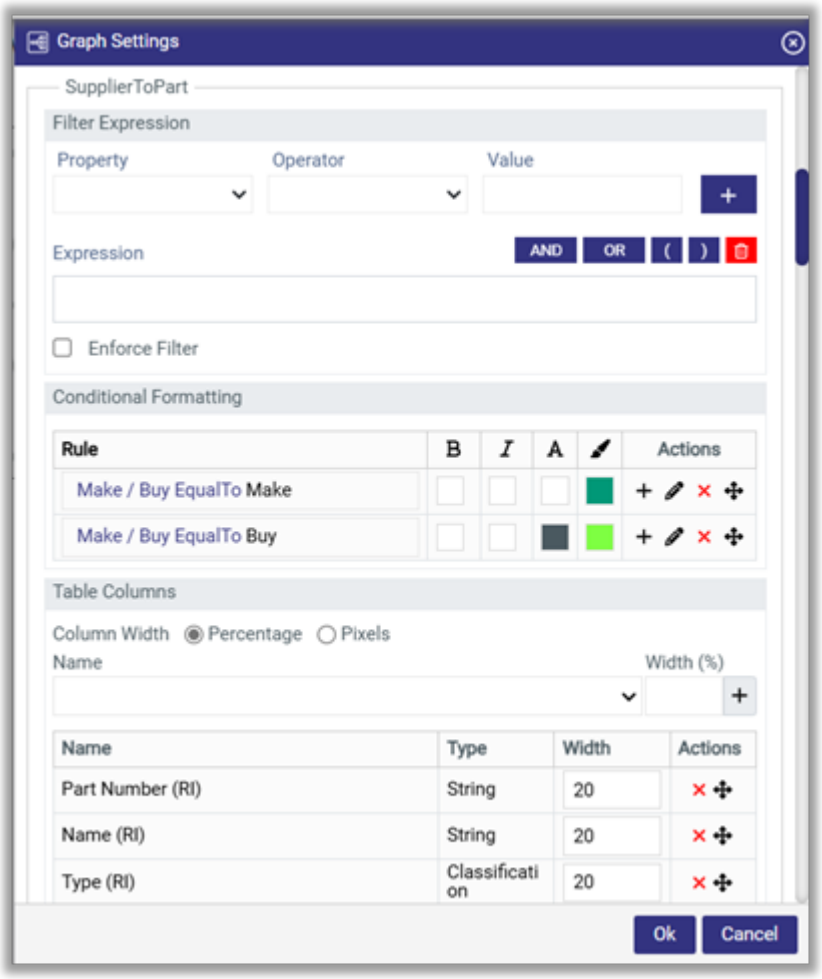
Graph Control

Graph control is used to show structure data of the context item in the graphical structure. Graph control is bound to a Query Definition, that defines which RelationshipTypes, RelatedItemTypes and properties to show in the control. By double clicking any node in the Graph, a flyout will be shown with the selected item properties along with its relationships. Graph control comes with two modes, read-only mode shows structure and properties, you can only navigate the structure and see properties of the selected node. You can also open the specific node in its own tab by selecting Open icon from the toolbar. In the edit mode, you can insert new nodes in the structure by selecting the Insert icon from the toolbar. In the InsertNode Flyout, allowed RelationshipTypes of the selected node are shown, by selecting specific RelationshipType and keyword, you can search and select a specific item to insert under the selected node. In edit mode, you can also edit the properties of the selected node. In order delete a specific node in edit mode, Delete icon can be selected from the toolbar. When you add Graph control on the page, it just shows only those Query Definitions defined with the same ItemType associated with the template.



- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the control. It can be localized by defined Languages and Locales inside innovator.
- **Query Definition**: It allows you to select the query that can be used to fetch data from Innovator database. It shows a list of all Query Definitions available in Innovator which are defined for the ItemType associated with the template.

- **Nodes**: It also allows you to configure the name shown in the GraphNode based on multiple properties. You can also specify font and background colors for the node at each relationship level. For each relationship, it adds a section to define filter expression, conditional formatting and columns that should be shown in the table at runtime.



- **Show Label**: It controls whether Label can be shown at the top of the Graph control.

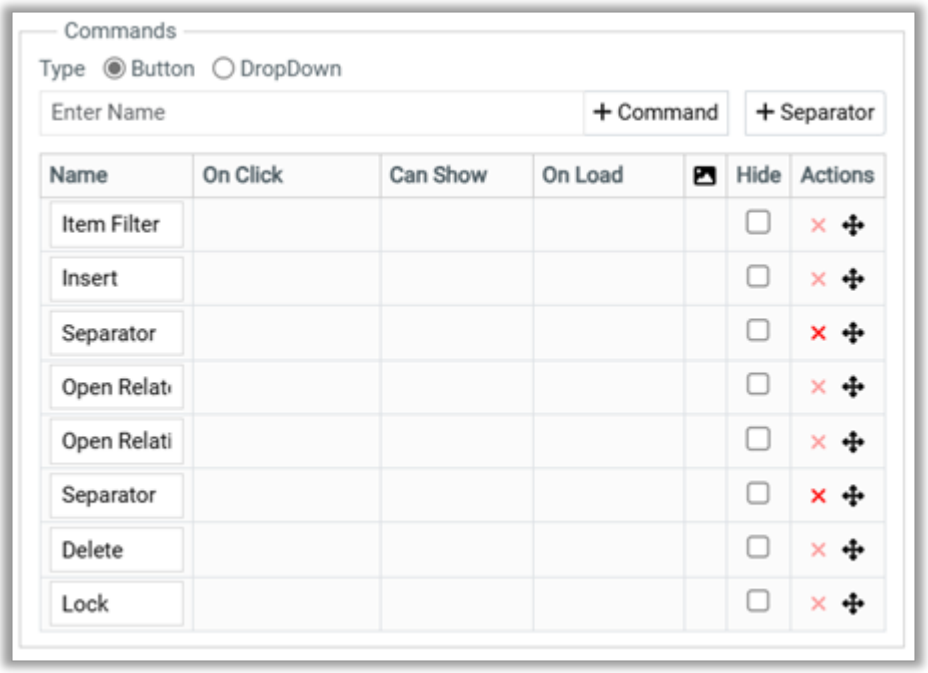
- **Commands**: Using Commands section, you can show or hide standard commands shown on the Graph toolbar. You can also define custom command by selecting Type as Button or DropDown and entering command name. For each custom command, you can set On Click, Can Show, On Load options.

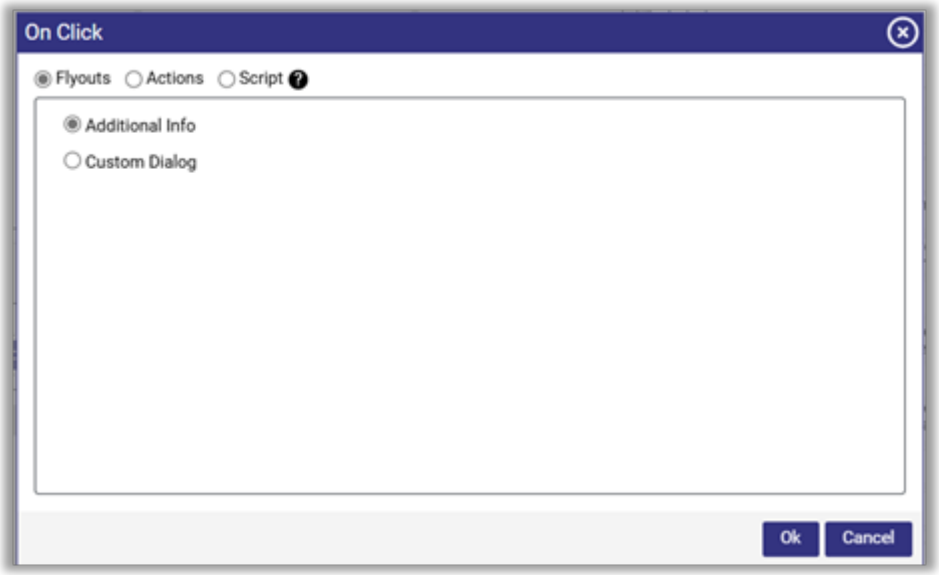
  Each custom command can be configured with ‘On Click’ handler to perform certain action. It can show a shortcut to the existing Innovator actions like Structure Browser, Where Used, Workflow Signoffs etc. It can also be configured to call a custom script. By selecting Script option in the dialog, you can associate custom script that contains code to interact with other Worksheet columns or controls in the Wizard. Script can also have code to interact with server by sending a request.

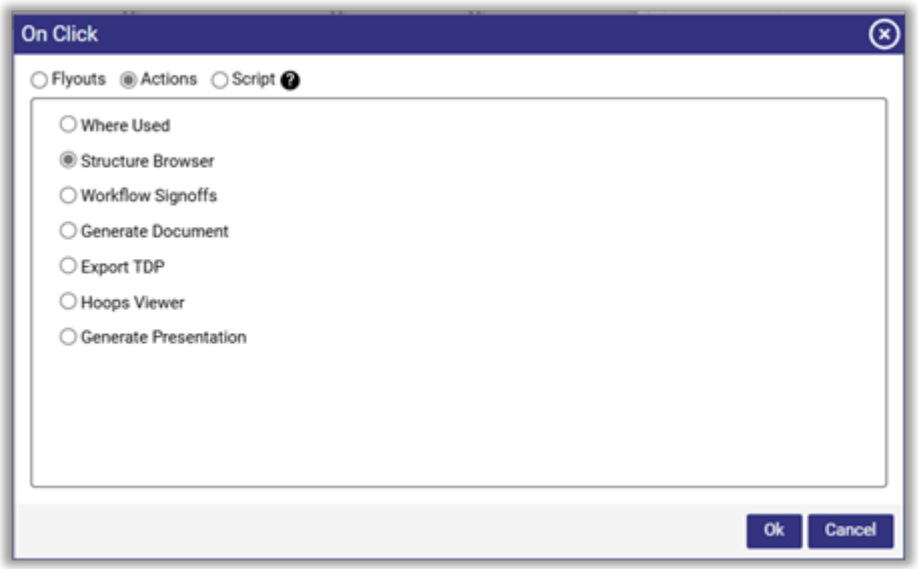
Each custom command can be configured with ‘Can Show’ option to define custom JavaScript that will return a boolean value. If returned value is true, command will be shown otherwise command will be hidden.

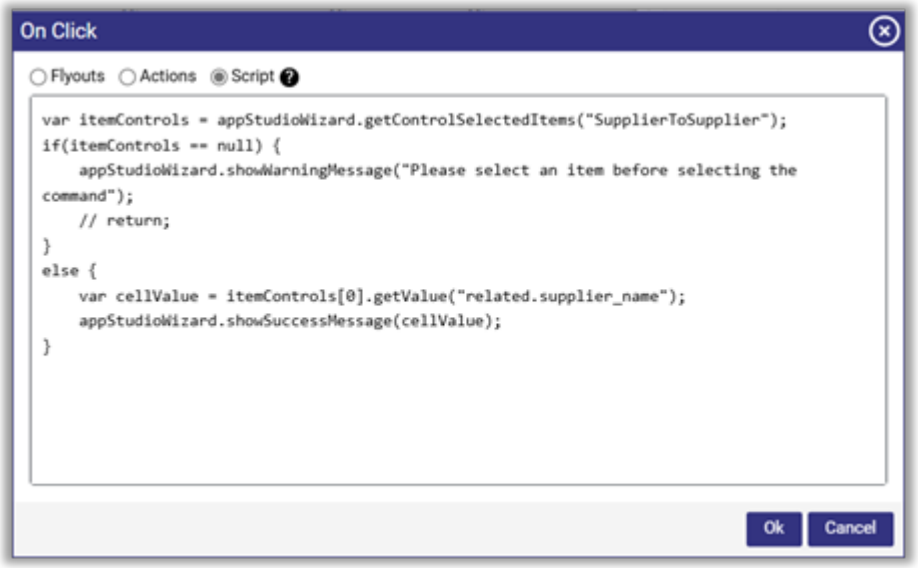
Each custom command of type ‘DropDown’ can be configured with ‘On Load’ option to load options for the DropDown at runtime.

For each command, you can configure the icon. You can also show or hide the command by selecting checkbox under Hide column.





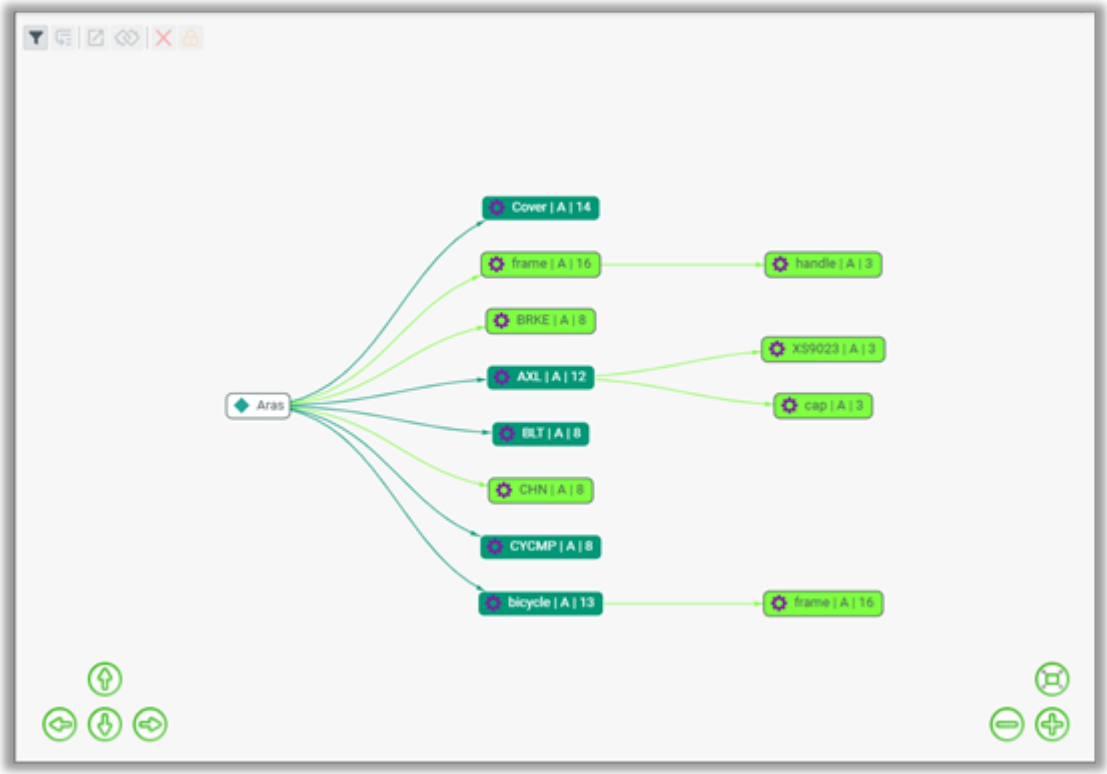




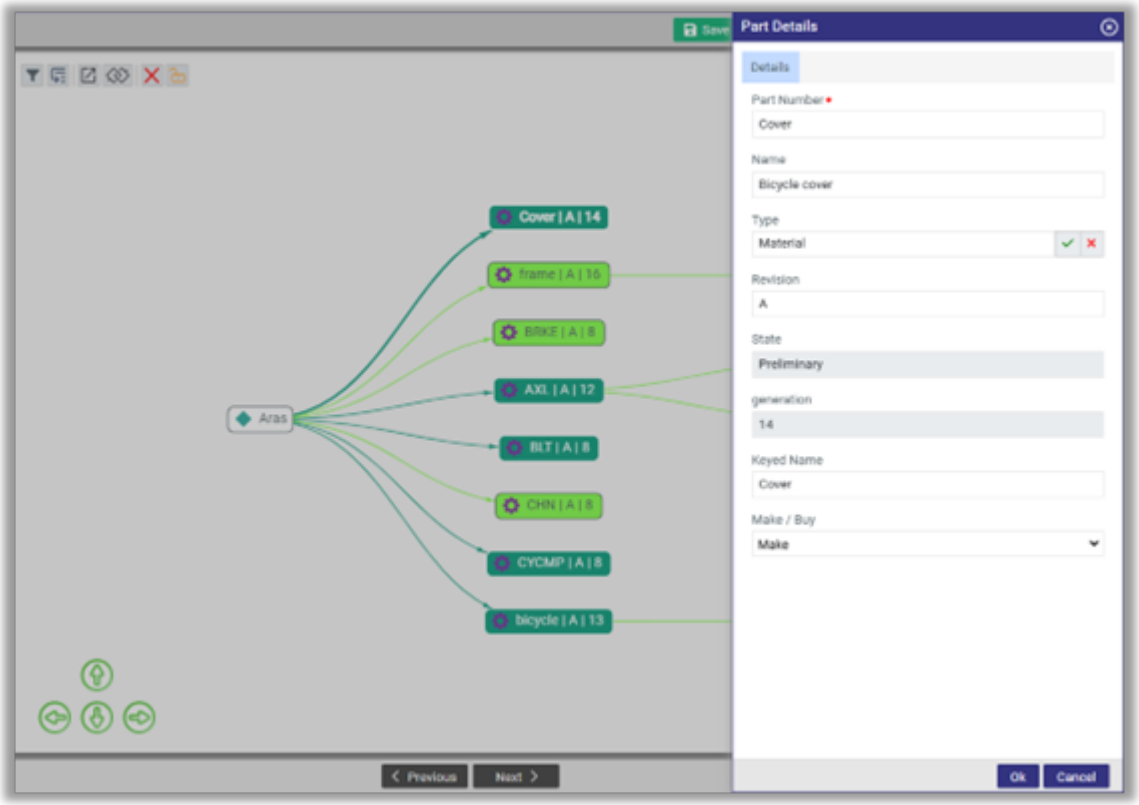
- **Height**: Graph control can be rendered in two different modes for the Height.
  - **Height**: Explicit
    In this mode, Max Height' should be supplied to limit the height of the control in the page. Control auto grows until it reaches 'Max Height', after that it shows scrollbar.
  - **Height**: Automatic
    In this mode, control auto grows until it reaches available page height, after that it shows scrollbar for the body. This mode can be set on Graph control only when the page that contains Graph control has only simple form fields apart from this Graph control.
- **Orientation**: It allows you to set the orientation for the graph structure shown in the control. Orientation can be horizontal or vertical.
- **Node Details Flyout**: It allows you to set the NodeDetails flyout width in percentage.

- **Insert Flyout**: It allows you to set the flyout width in percentage. It also allows you to show Add command in the toolbar in the insert flyout. Using this command, new item can be created on the fly as part of the insert operation.
- **Access**: It allows you to define roles which can access control. Roles (in the form group identities) can be at Show and Edit level. When control is hidden because you do not have defined role, data associated with this control is not even fetched from server.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. If Read Only is selected as ‘Yes’, you can only navigate structure, and when the node is double clicked its properties are shown in the flyout. If Read Only is selected as ‘No’, along with the navigation, you can also edit the structure as well as properties of the selected node. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only. In order to see graph control in edit mode, you need to put context item in edit mode.

    - Read Only: Yes



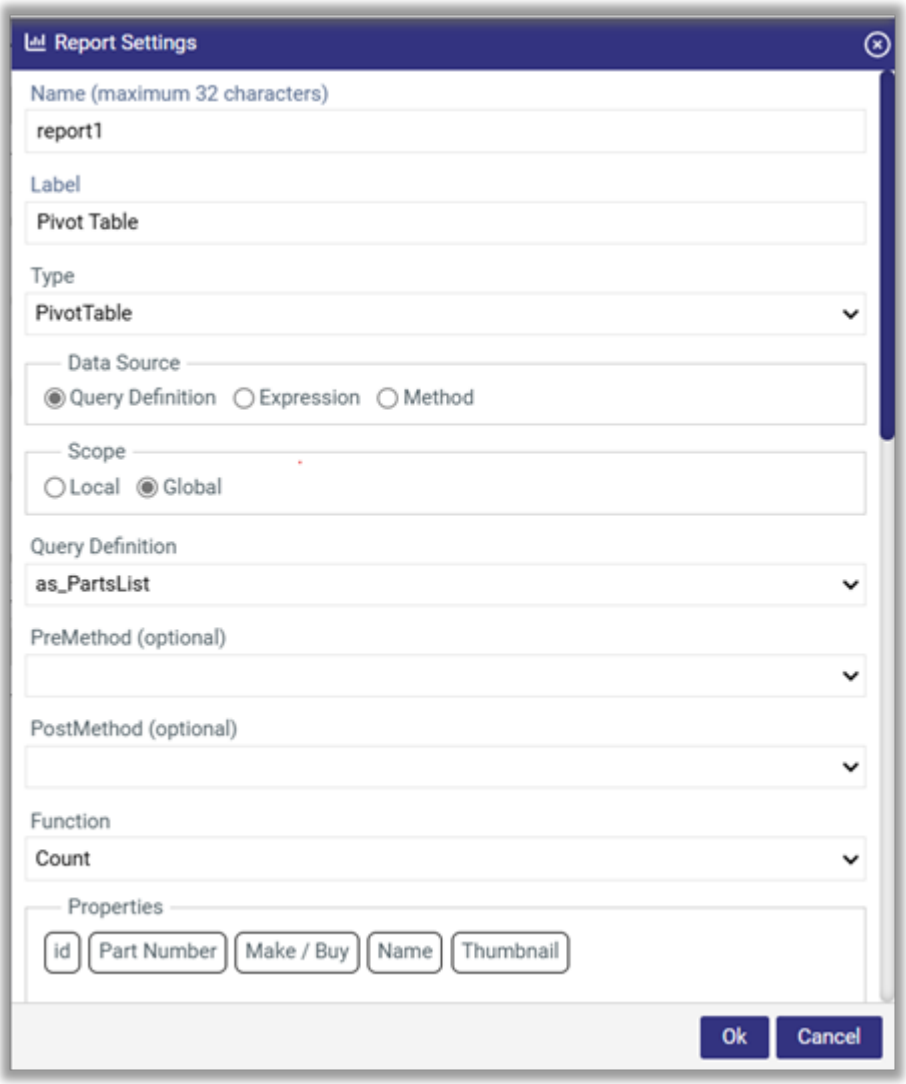
    - Read Only: No



- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Font**: It allows you to set the required font and size for the node names and form field values of the Graph control.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **On Load**: OnLoad event allows you to write custom logic that will be executed after the data in the control is loaded. Please refer to the Template Customization section for more details.
- **On Insert**: On Insert event will be launched after Item has been inserted in the Graph control. Event handler will receive inserted item details via eventData, this object will be available in the scope of the event handler code. Please refer to the Template Customization section for more details.
- **On Select**: On Select event will be launched after selecting the item in the Graph control. Event handler will receive selected item details via eventData, this object will be available in the scope of the event handler code. Please refer to the Template Customization section for more details.

Report Control

Report control is used to generate reports from the available data. Data will be fetched for a report control based on the Query Definition selected for the control. Report can be generated based on the data associated with the context item, or it could be generated based on global data of any other ItemType. Report control allows you to present data in form of PivotTable, Line Chart, Bar Chart, Area Chart etc.



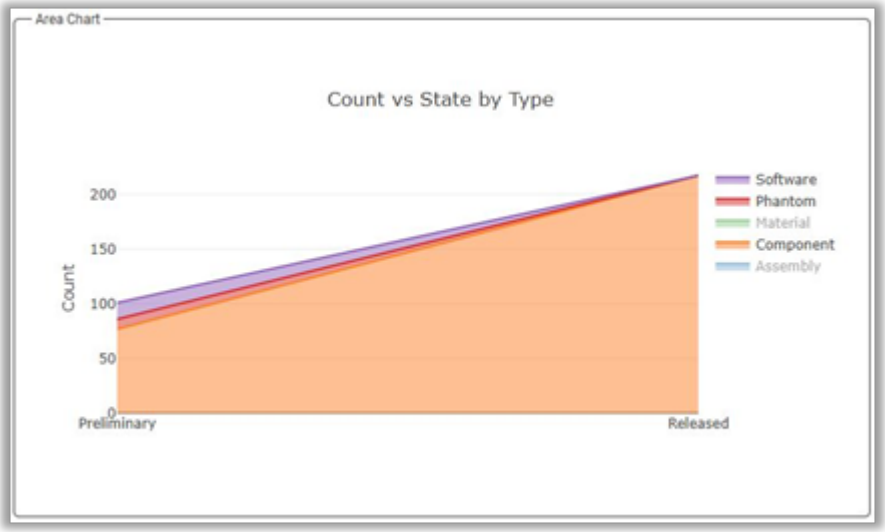
- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the control. It can be localized by defined Languages and Locales inside innovator.

- **Type**: It allows you to select the type of the chart for presenting the report data. Its dropdown shows list of available report types like PivotTable, Table BarChart, Heatmap, Row Heatmap, Column Heatmap, Line Chart, Bar Chart, Stacked Bar Chart, Area Chart, Stacked Area Chart.

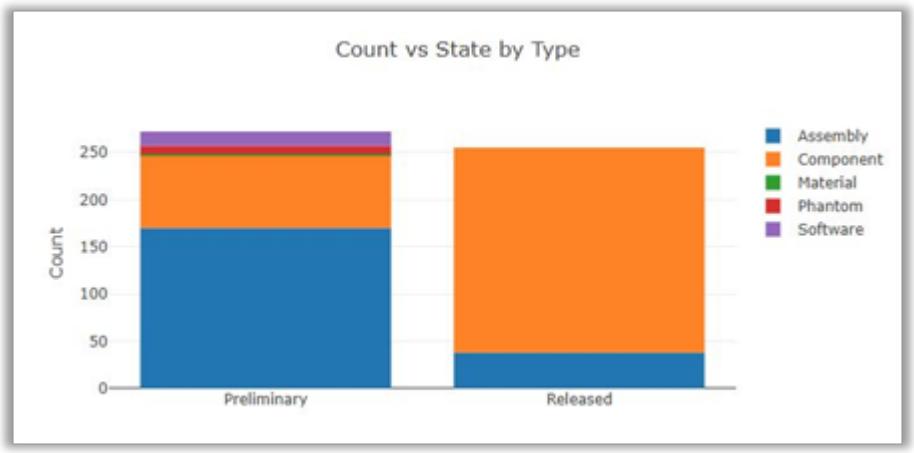
  - Type: PivotTable

Type	State	Preliminary	Released	Totals
Assembly		182	36	218
Component		97	197	294
Material		2		2
Phantom		9		9
Software		14		14
Totals		304	233	537

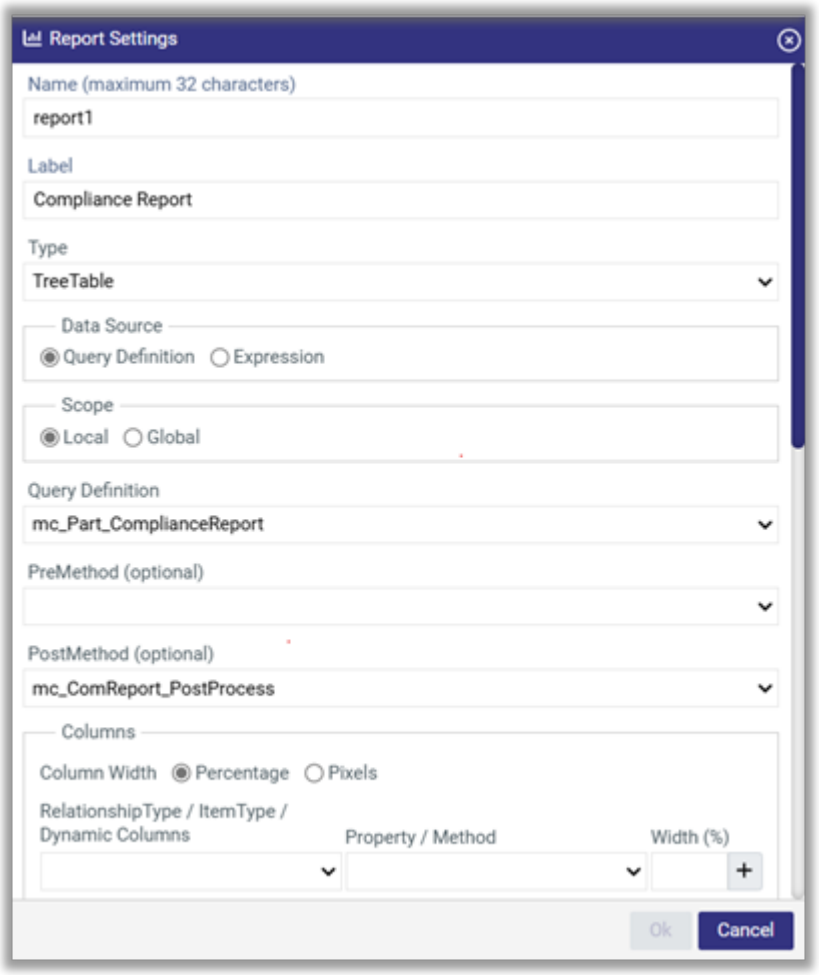
  - Type: Area Chart

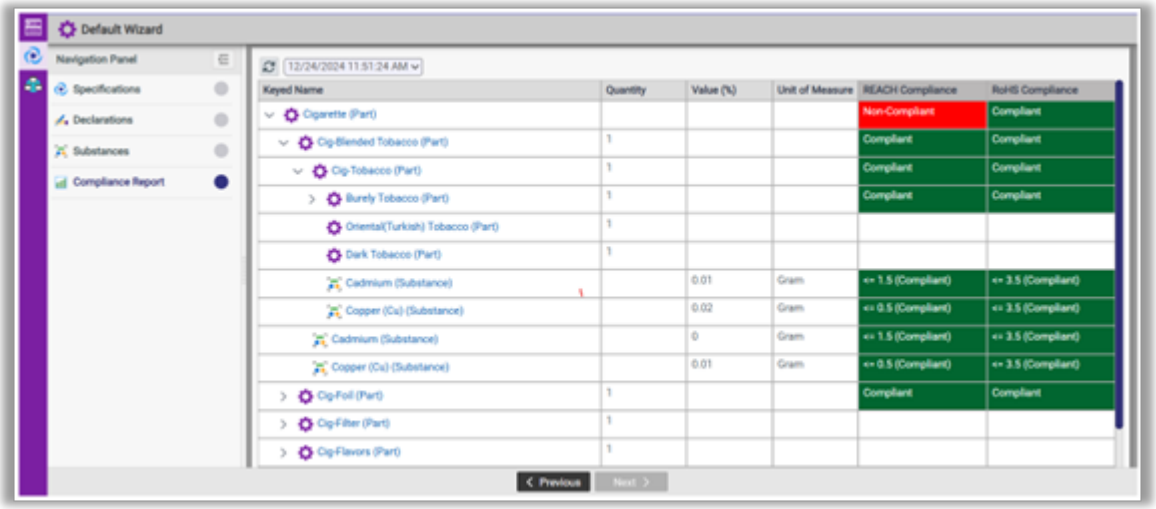


  - Type: Stacked Bar Chart



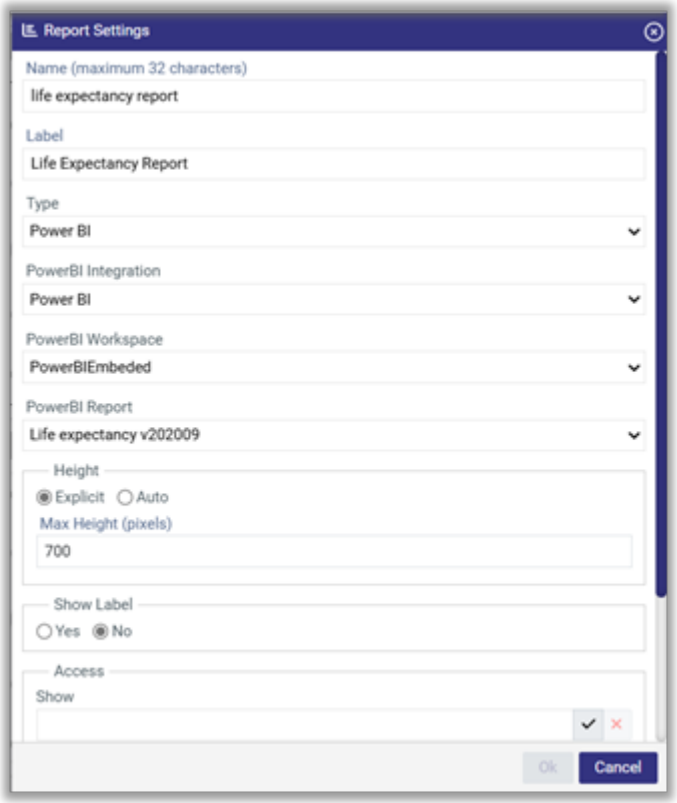
- Type: TreeTable
  TreeTable report is used to show report data in the form of table or tree-table. The Report will be table or tree-table based on data returned from the configured Query Definition. TreeTable Report settings show section to configure columns that you want to see on the report. Columns can be configured based on the properties returned from Query Definition. TreeTable report can also support dynamic columns. To configure dynamic columns, you need to select DynamicColumns option from the first column dropdown. When you select DynamicColumns option, second column shows list of ServerMethods. Dynamic columns are defined based on columns definition values returned from configured ServerMethod. When you select DynamicColumns for the report, report PostMethod setting should be configured. Configured ServerMethod under PostMethod setting, should return values that need to be shown under report DynamicColumns.

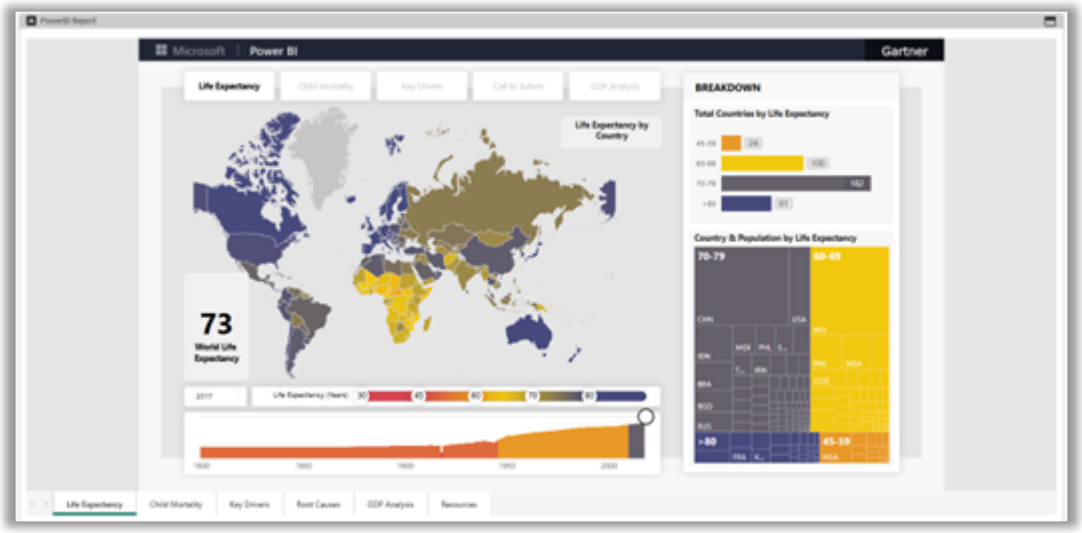




- Type: Power BI

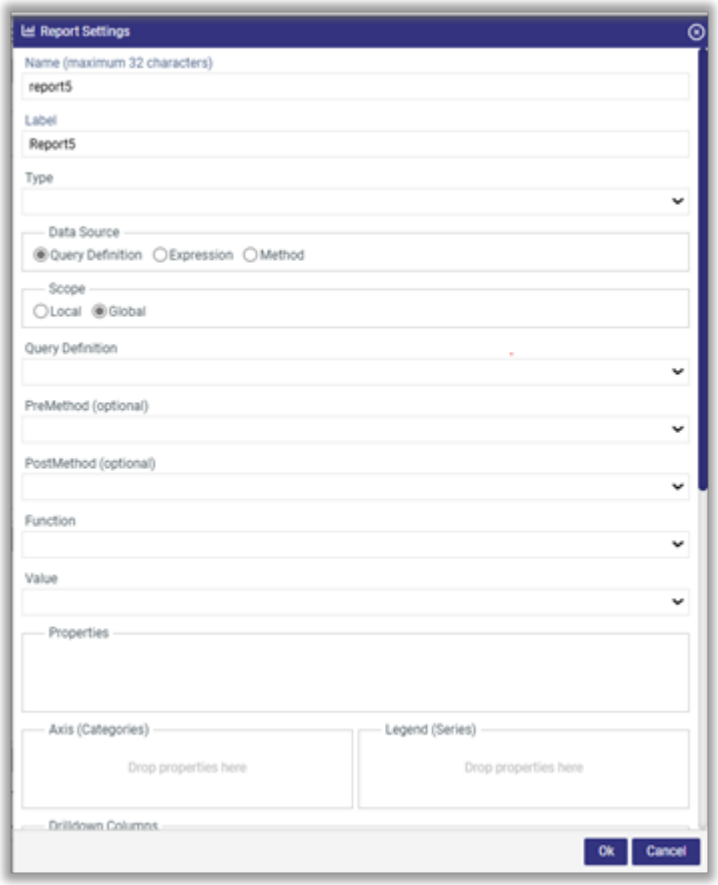
  Power BI reports can be accessed from Power BI server through the Power BI Integration configured using Integration Framework. Please refer Power BI Integration documentation on how to configure the integration. In order to create Power BI report, integration item should be created under Integration Framework. All available Power BI integrations are shown under PowerBI Integration dropdown in Power BI Report settings. Since reports in Power BI server are organized in workspaces, after selecting integration item available workspaces will be shown under PowerBI Workspace dropdown. Upon selecting the workspace, available reports from that workspace will be shown under PowerBI Report dropdown.



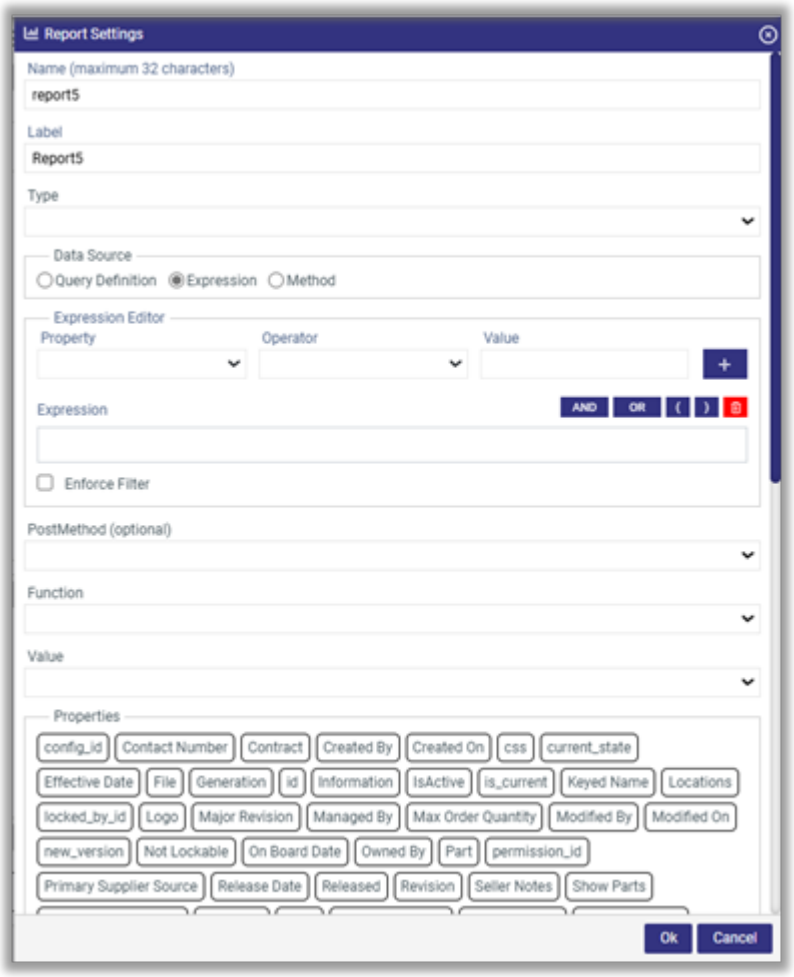


Data Source: Data source for the report can be ‘Query Definition’, ‘Expression’, and ‘Method’.

- **Data Source:** Query Definition
  In order to create a report based on the Query Definition, Query Definition has to be created that accepts id for local report or no-id for global report scope. When you select his option, you will get see settings like Scope, Query Definition, PreMethod, and PostMethod.

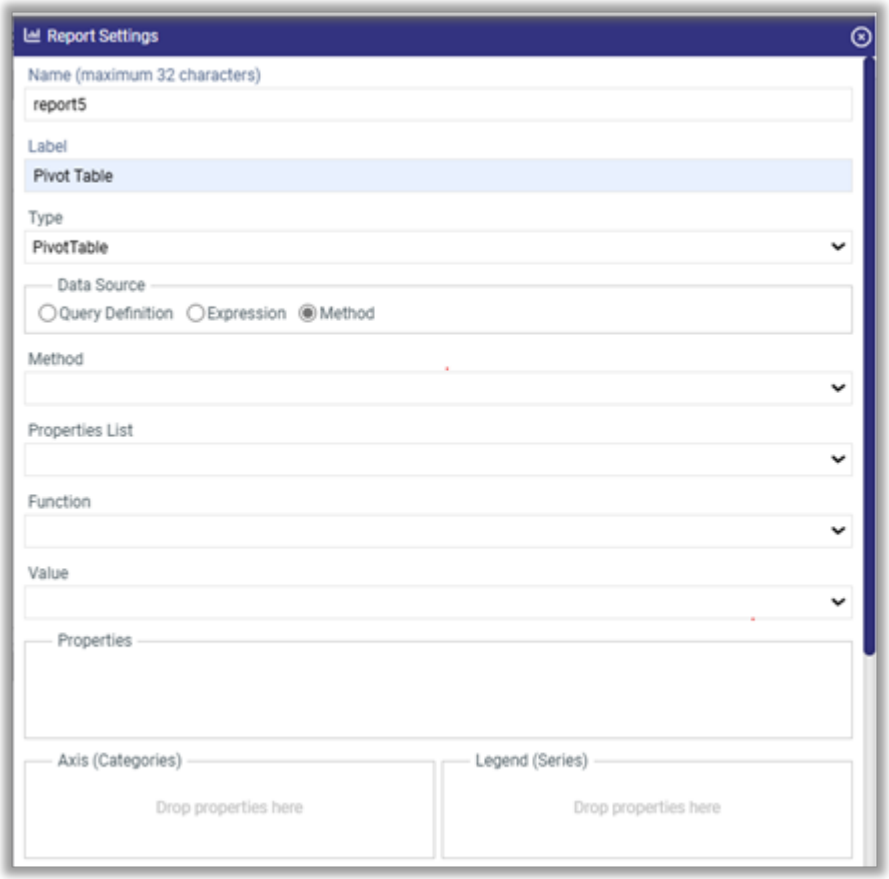


- **Data Source:** Expression
  When you select option ‘Expression’, expression editor will be shown with all properties from the context ItemType. Complex filter expression can be built with grouping and logical operators. At runtime, this filter expression will be converted to AML and will be executed to get results from server. With this option you will get see setting PostMethod, using that you can further filter results based on custom logic.



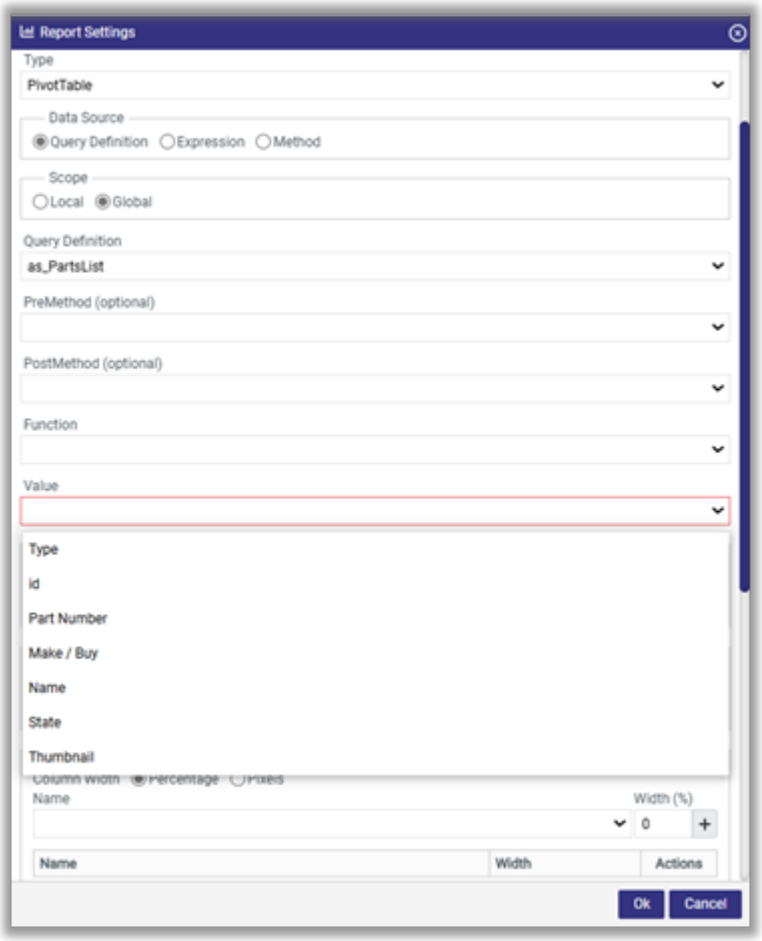
- **Data Source:** Method

  When you select option ‘Method’, it will show you the option to select ServerMethod, that can contain logic to return list of ids, that can be used to generate the report. To specify which properties are returned from the ServerMethod, you can define a List item in Aras Innovator with a list of all properties names as options. That list can be selected for setting ‘Properties List’.

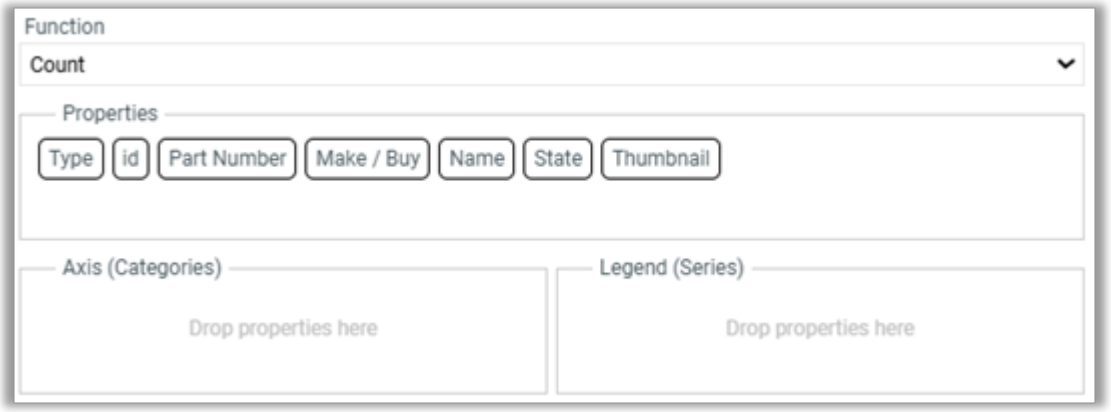


- **Scope**: It allows you to define the report with Local and Global scopes. If you select Global scope, it will show all the Query Definitions which are available in Innovator. With Local scope, it will only show Query Definitions defined based on the ItemType associated with the current template.
- **Query Definition**: It allows you to select the query that can be used to fetch data from Innovator database. It shows a list of all Query Definitions available in Innovator based on the selected Scope.
- **PreMethod**: It is an optional setting, in the absence of this, it will always pass context item id to the Query Definition to get the report data. By setting this value, it will allow you to pass different item id to QueryDefintion execution in place of the context item id. Upon clicking the dropdown, you will get to see list of all server methods. You need to select an appropriate server method that takes the context item id as an input, transform it to some other relatedItem id, and returns transformed id.
- **PostMethod**: It is an optional setting, in the absence of this, it will always return the json generated from the Query Definition execution as the report data. By setting this value, it will allow you to transform json returned from the QueryDefintion execution. Upon clicking the dropdown, you will get to see list of all server methods. You need to select an appropriate server method that takes the json returned from QueryDefintion execution as an input, transform it to different json that is suitable to render inside the report control, and returns transformed json.
- **Function**: It allows you to select appropriate aggregate function that will be applied on the report data returned from the QueryDefintion execution. Upon clicking the dropdown, you will

get to see list of all standard aggregate functions. If the selected Function is other than Count, you need to select a property for Value in order to apply the function.

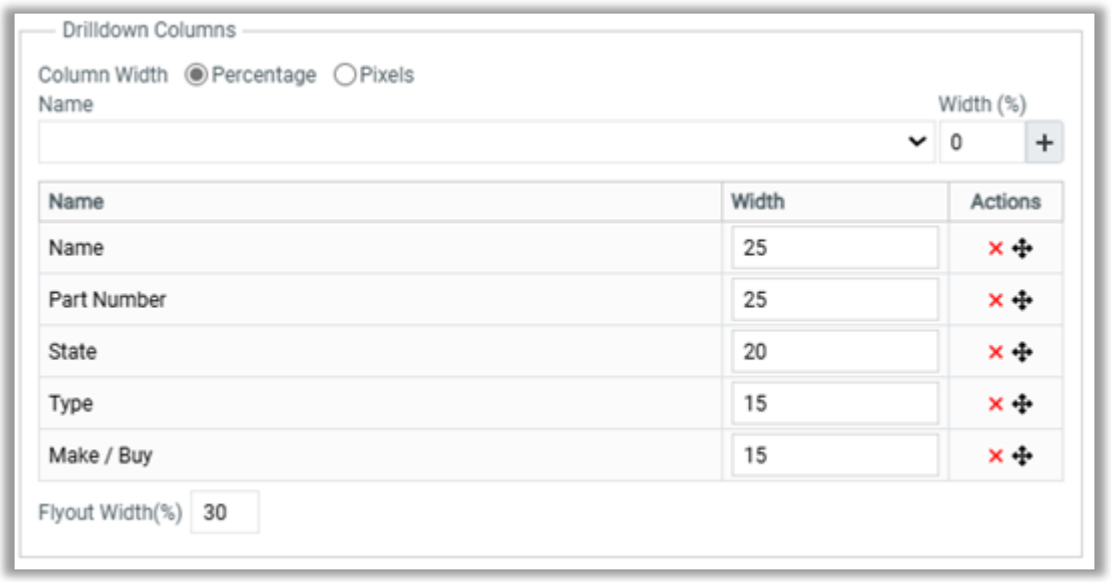


- **Properties**: It shows you the list of properties that can be returned from the Query Definition execution. While defining Query Definition, you need to make sure all required properties for the report are returned from the Query Definition. You can place these properties under Group By and Column settings with a simple drag and drop operation.



- **Axis (Categories)**: It allows you to select properties for grouping and sub-grouping in the horizontal direction.
- **Legend (Series)**: It allows you to select properties for grouping and sub-grouping in the vertical direction.

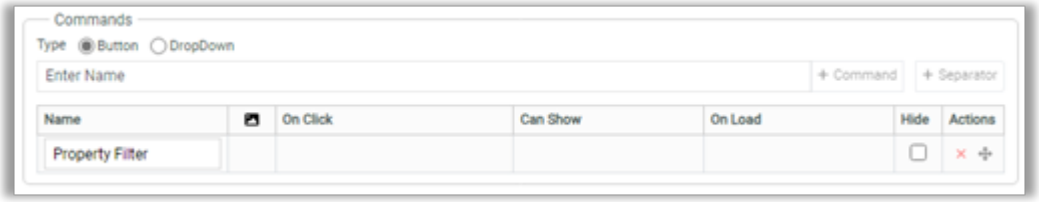
- **Drilldown Columns**: Clicking on the numbers shown in the reports, Drilldown Report will be shown in the Flyout with all the relevant items. Using DrilldownColumns settings section, you can add all the required columns to the Drilldown Flyout. Columns can be added based on the properties returned from the QueryDefintion. Using Column Width option, you can set column widths in percentage or pixels. If you select Column Width as Percentage, you never see horizontal scrollbar, widths of the columns will get adjusted to the available total width of the tree-table. If Column Width option is selected as Pixels, you will see horizontal scrollbar if the combined widths of all columns is more than available flyout width. As part of adding column, you need to select the property from QueryDefinition, to which you want to bind the column. After adding columns, you can edit the Width of the column, or you can reorder column using drop-drop icon from the Actions columns. You can delete existing column by selecting delete icon from the Actions column. You can also define the width of the Flyout in percentage with respect to main page.



- **Show Label**: It controls whether Label can be shown at the top of the report control.
- **Report Data Cache**: It allows caching the report data to render the report quickly next time when it is opened. If option 'None' is selected, report data will never be cached, it renders report based on current state of the database. If option 'User' is selected, report data is cached for each user when the report is rendered for the first time. Cached report data for one user, can never be shown to the other user. If option 'Global' is selected, when the report is rendered for any user first time, its data is cached, and the same data will be shown to other users.

- **Commands**: Using Commands section, you can show or hide standard commands shown on the Table toolbar. You can also define custom command by selecting Type as Button or DropDown and entering command name. For each custom command, you can set On Click, Can Show, On Load options.
  Each custom command can be configured with 'On Click' handler to perform certain action. It can show a shortcut to the existing Innovator actions like Structure Browser, Where Used, Workflow Signoffs etc. It can also be configured to call a custom script. By selecting Script option in the dialog, you can associate custom script that contains code to filter the report data based on the filter expression or parameters.
  Each custom command can be configured with 'Can Show' option to define custom JavaScript that will return a boolean value. If returned value is true, command will be shown otherwise command will be hidden.
  Each custom command of type 'DropDown' can be configured with 'On Load' option to load options for the DropDown at runtime.

For each command, you can configure the icon. You can also show or hide the command by selecting checkbox under Hide column.



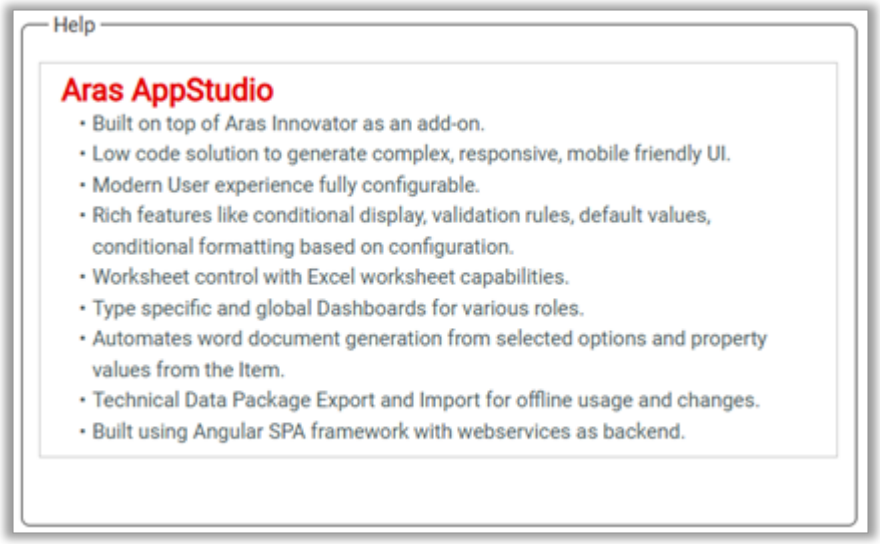
- **Placement**: It allows you to place the report in the available horizontal space through alignment (left, right, center) setting. It also allows you to set maximum height that can be used to show report data.
- **Access**: It allows you to define the roles which can access control. You can configure Roles (in the form group identities) to whom this control should be shown.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.

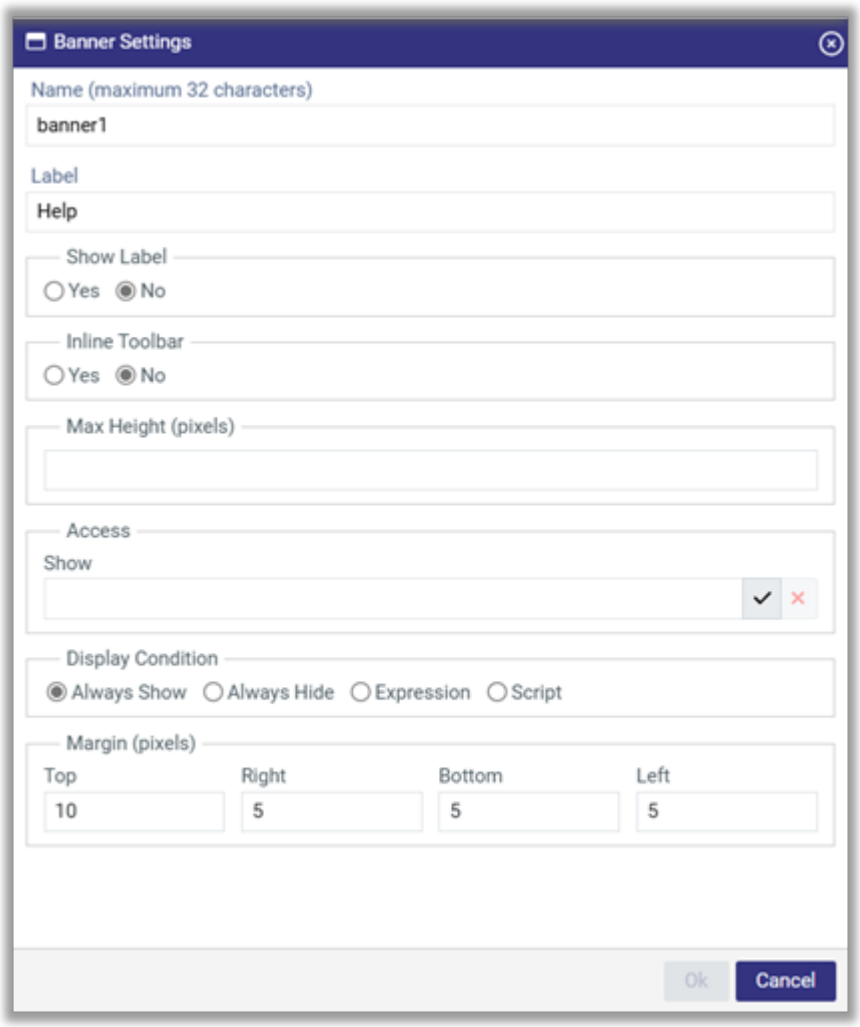
Runtime View:

Drilldown Report:

Banner Control

Banner control can be used to show static content like instructions or help on the page. Banner control with static content is added through rich-text editor inside the template. At runtime, it just shows rich-text to the user.





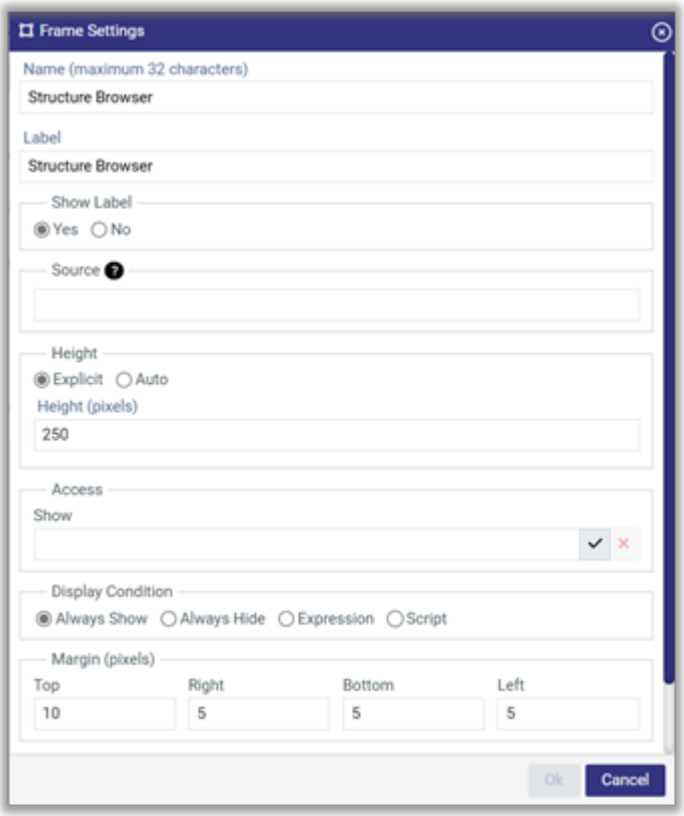
- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the control. It can be localized by defined Languages and Locales inside innovator.
- **Show Label**: It controls whether Label can be shown at the top of the banner control.
- **Inline Toolbar**: It allows you to show rich-text editor right in the page with its toolbar to edit the content. If its value is ‘No’, rich-text editor is shown in the pop-up upon clicking on the icon from the banner control.
- **Max Height**: It defines maximum height that can be used to show the banner control on the page. In the absence of this value, banner control is expanded based on its static content.
- **Access**: It allows you to define the roles which can access control. You can configure Roles (in the form group identities) to whom this control should be shown.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property

values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.

- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.

Frame Control

Frame control is used to show content from the external page. It can be used inside the page to show a report generated from the external BI tool.

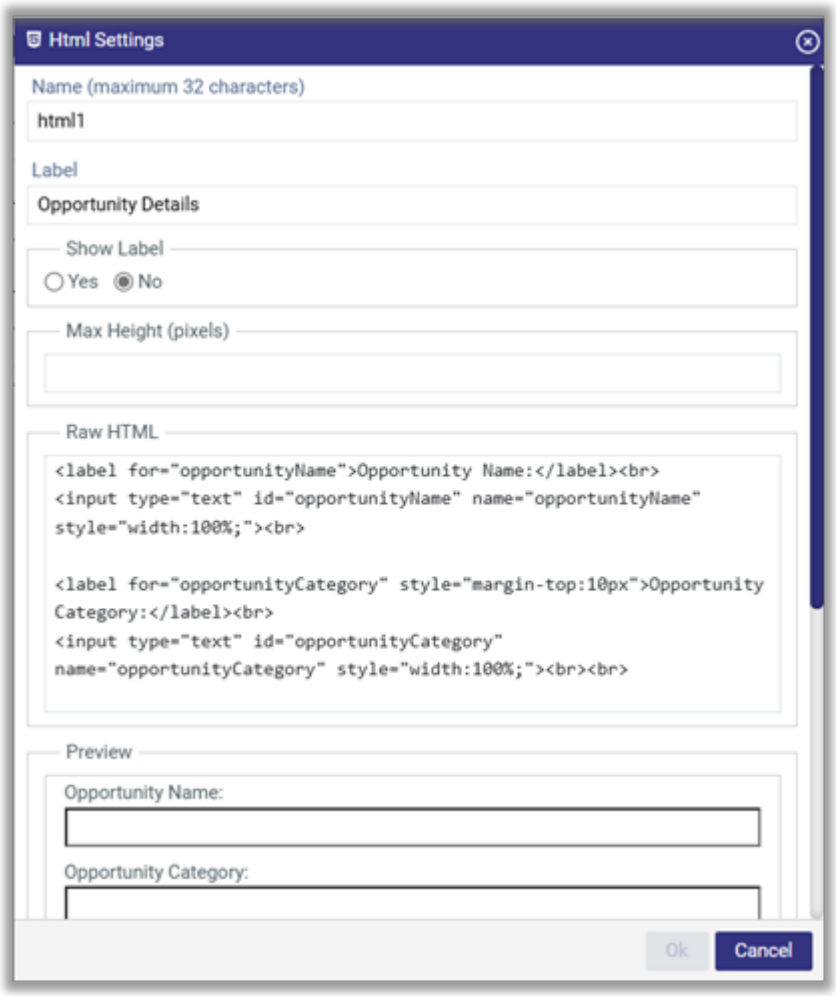


- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the control. It can be localized by defined Languages and Locales inside innovator.
- **Show Label**: It controls whether Label can be shown at the top of the Frame control.
- **Source**: It allows you to set URL of the external page. URL can contain variables that represent properties (system & custom) of the context item.
  https://www.aras.com?id=<%id%>&itemType=<%itemType%>&classification=<%classification%>
- **Height**: Frame control can be rendered in two different modes for the Height.
  - **Height**: Explicit
    In this mode, 'Max Height' should be supplied to limit the height of the frame in the page. Frame auto grows until it reaches 'Max Height', after that it shows scrollbar for the body of the frame.
  - **Height**: Auto
    In this mode, the frame controls auto grows until it reaches available page height, after that it shows scrollbar for the body of the frame.

- **Access**: It allows you to define the roles which can access control. You can configure Roles (in the form group identities) to whom this control should be shown.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **OnLoad**: OnLoad event allows you to write custom logic that will be executed after the data in the control is loaded. It allows you to write a custom script to load dynamic content in the Frame control. Please refer to the Template Customization section for more details.

HTML Control

HTML control is used to show custom UI in the pages and dialogs. HTML content added to this control will be rendered in the same DOM as of the Aras ProAppDesigner Wizard. This control supports only inline styling for the content.

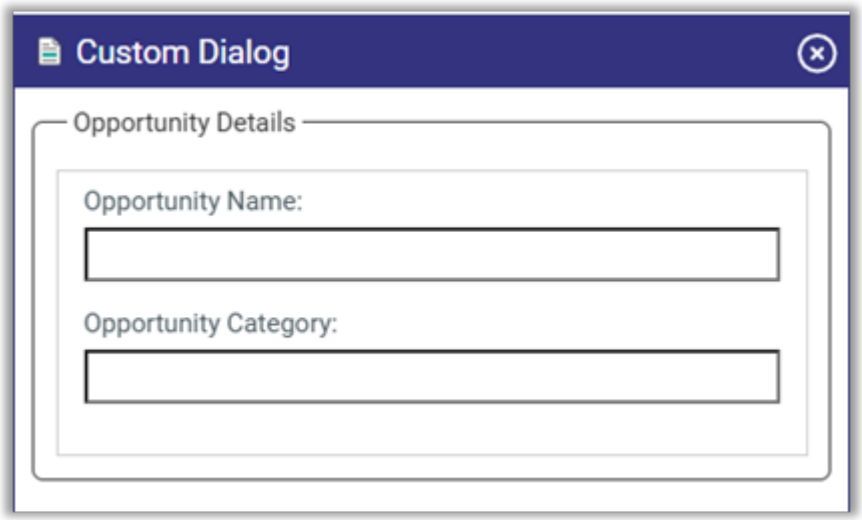


- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the control. It can be localized by defined Languages and Locales inside innovator.
- **Show Label**: It controls whether Label can be shown at the top of the HTML control.
- **Max Height**: It defines maximum height that can be used to show the HTML control on the page. In the absence of this value, HTML control is expanded based on its static content.
- **Raw HTML**: It allows you to write custom HTML along with the inline styles.

```
 <label for="opportunityName">Opportunity Name:</label><br>
 <input type="text" id="opportunityName" name="opportunityName"
style="width:100%;"><br>
 <label for="opportunityCategory" style="margin-top:10px">Opportunity
Category:</label><br>
 <input type="text" id="opportunityCategory"
name="opportunityCategory" style="width:100%;"><br><br>
```

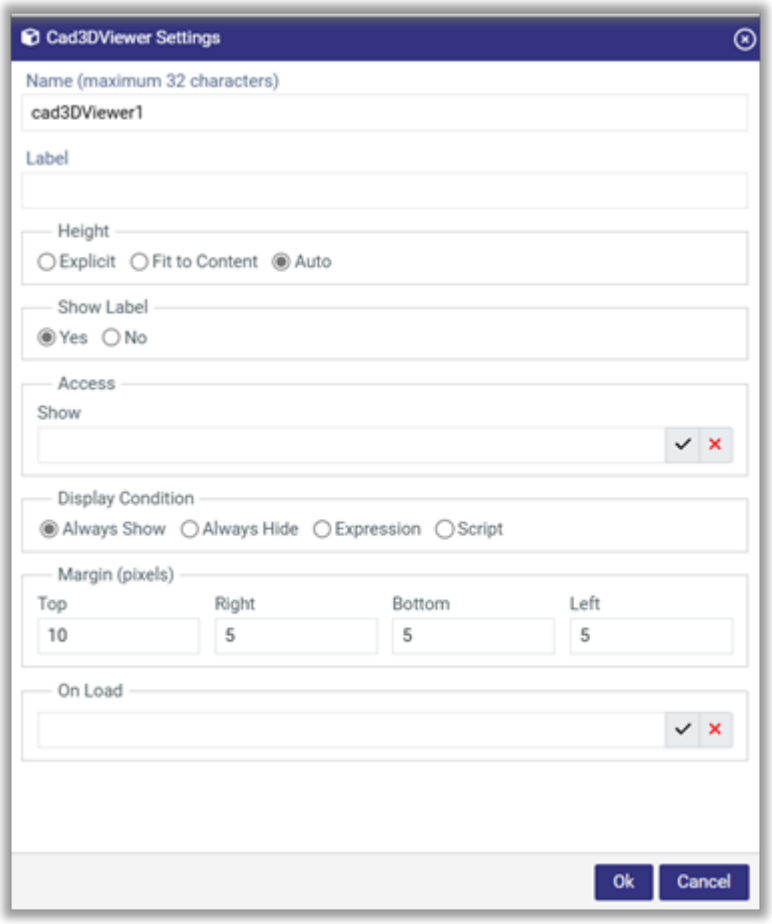
- **Preview**: Preview area shows runtime view of the custom HTML entered as you make changes to the Raw HTML.
- **Access**: It allows you to define the roles which can access control. You can configure Roles (in the form group identities) to whom this control should be shown.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **OnLoad**: OnLoad event allows you to write custom logic that will be executed after the data in the control is loaded. It allows you to write a custom script to assign default values to the fields shown in the HTML control. The custom script can also associate events with the fields shown in the custom UI in order to apply some dynamic logic with the UI. Please refer to the Template Customization section for more details.

Runtime View:



CADViewer Control

3DViewer control is used to render CAD viewable associated with a CAD Document item.

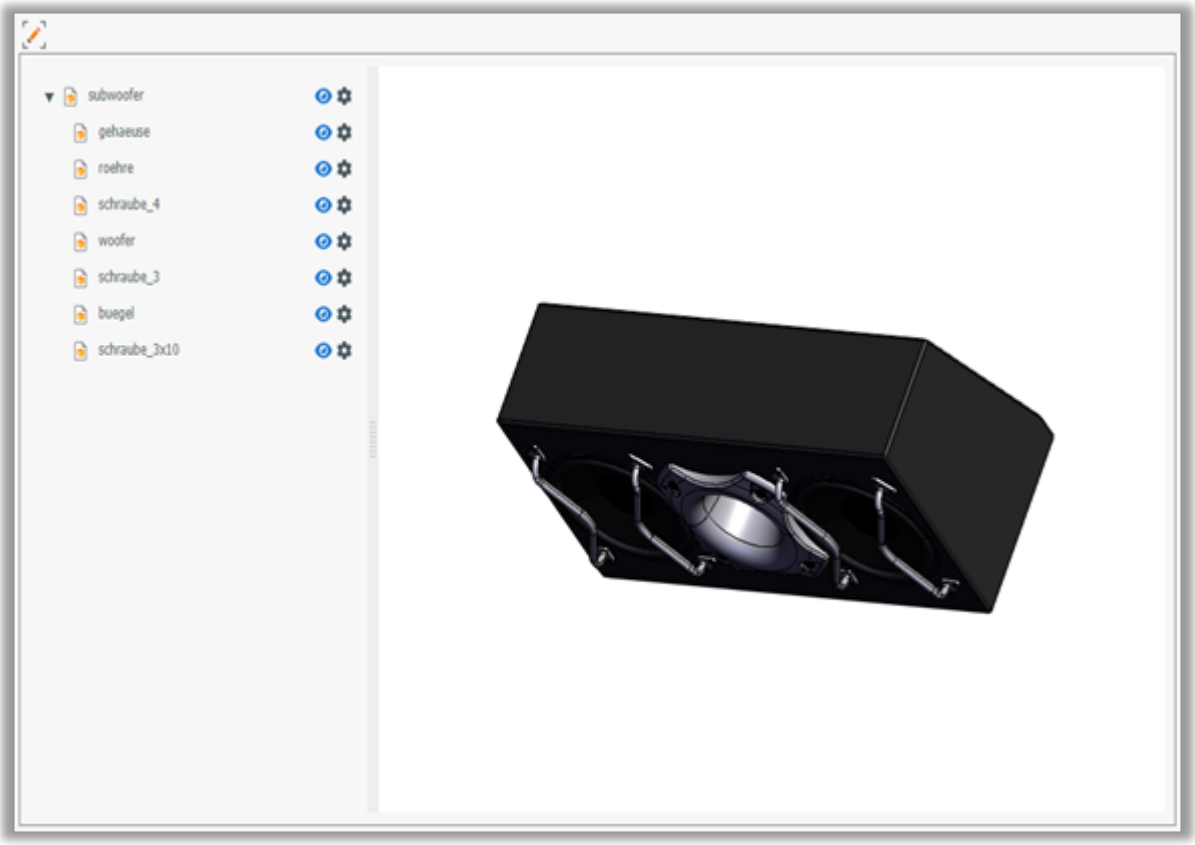


- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the control. It can be localized by defined Languages and Locales inside innovator.
- **Height**: CADViewer control can be rendered in three different modes for the Height.
- **Height**: Explicit
- In this mode, 'Max Height' should be supplied to limit the height of the control in the page. Control auto grows until it reaches 'Max Height', after that it shows scrollbar.
    - **Height**: Fit to Content
      In this mode, CADViewer control grows its height without showing any scrollbar. If it doesn't get required height by the page, then page will start showing scrollbar.
    - **Height**: Auto
      In this mode, control auto grows until it reaches available page height, after that it shows scrollbar for the body. This

mode can be set on the CADViewer control only when the page that contains CADViewer control has only simple form fields apart from this CADViewer control.

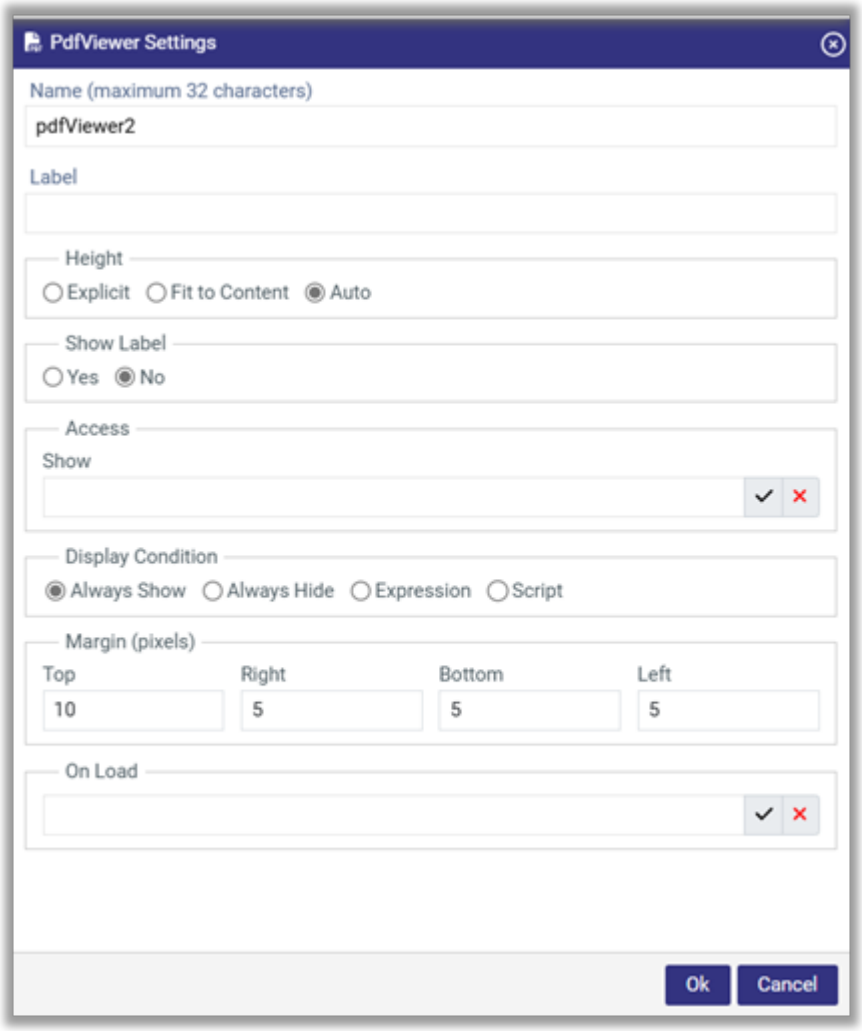
- **Show Label**: It controls whether Label can be shown at the top of the CADViewer control.
- **Access**: It allows you to define the roles which can access control. You can configure Roles (in the form group identities) to whom this control should be shown.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Font**: It allows you to set the required font and size for the control.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **OnLoad**: OnLoad event allows you to write custom logic that will be executed after the data in the control is loaded. It allows you to write a custom script to find and assign CADDocument id to the CADViewer control. Please refer to the Template Customization section for more details.

Runtime View:



PDFViewer Control

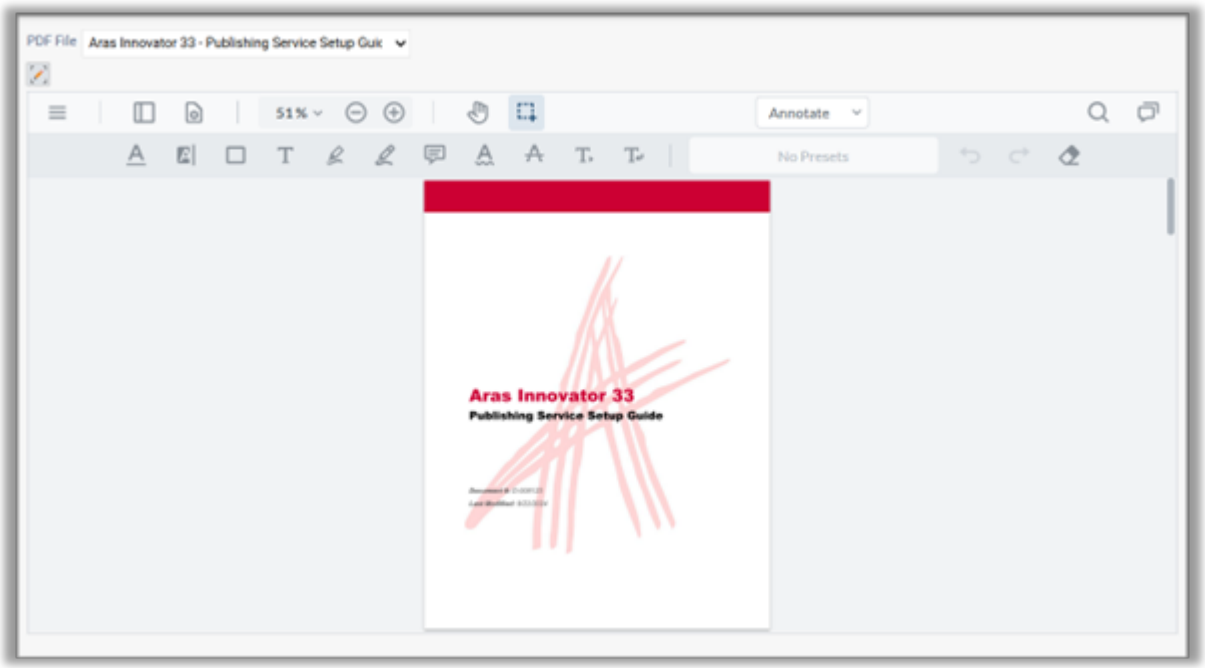
PDFViewer control is used to render PDF documents associated with a Document item.



- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the control. It can be localized by defined Languages and Locales inside innovator.
- **Height**: PDFViewer control can be rendered in three different modes for the Height.
    - **Height**: Explicit
      In this mode, 'Max Height' should be supplied to limit the height of the control in the page. Control auto grows until it reaches 'Max Height', after that it shows scrollbar.
    - **Height**: Fit to Content
      In this mode, PDFViewer control grows its height without showing any scrollbar. If it doesn't get required height by the page, then the page will start showing scrollbar.

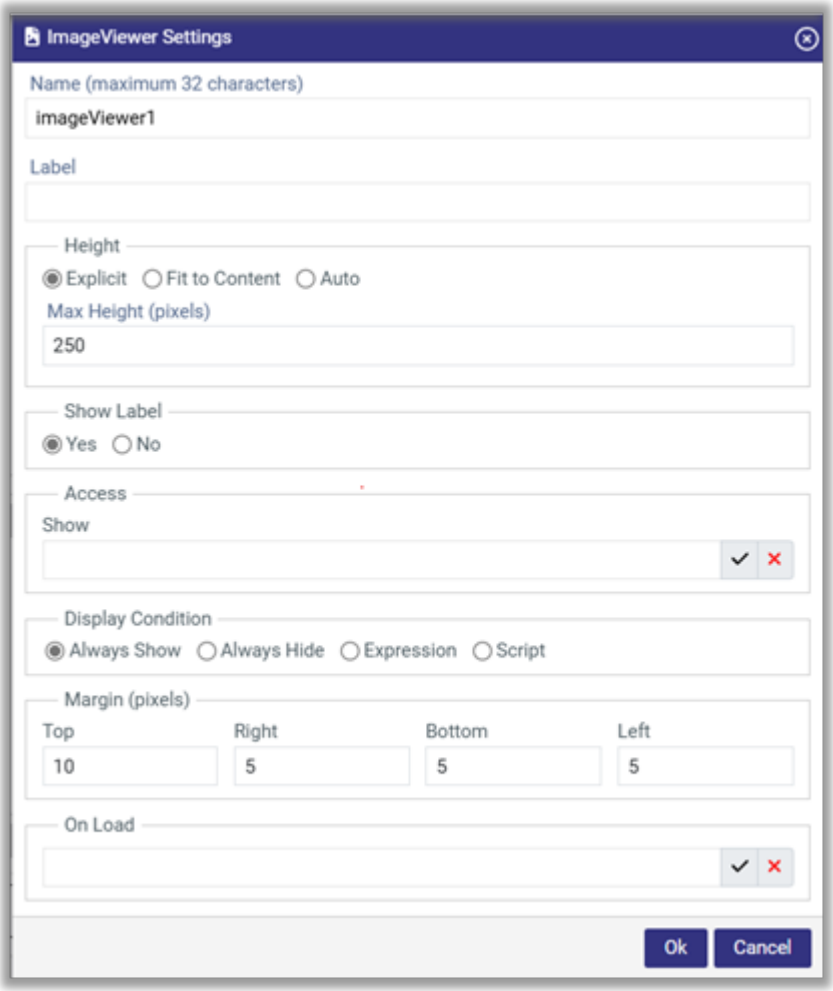
  - **Height**: Auto
    In this mode, control auto grows until it reaches available page height, after that it shows scrollbar for the body. This mode can be set on the PDFViewer control only when the page that contains PDFViewer control has only simple form fields apart from this PDFViewer control.
- **Show Label**: It controls whether Label can be shown at the top of the PDFViewer control.
- **Access**: It allows you to define the roles which can access control. You can configure Roles (in the form group identities) to whom this control should be shown.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If the returned value is false, the control will be hidden.
- **Font**: It allows you to set the required font and size for the control.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **OnLoad**: OnLoad event allows you to write custom logic that will be executed after the data in the control is loaded. It allows you to write a custom script to find and assign Document id to the PDFViewer control. Please refer to the Template Customization section for more details.

Runtime View:



ImageViewer Control

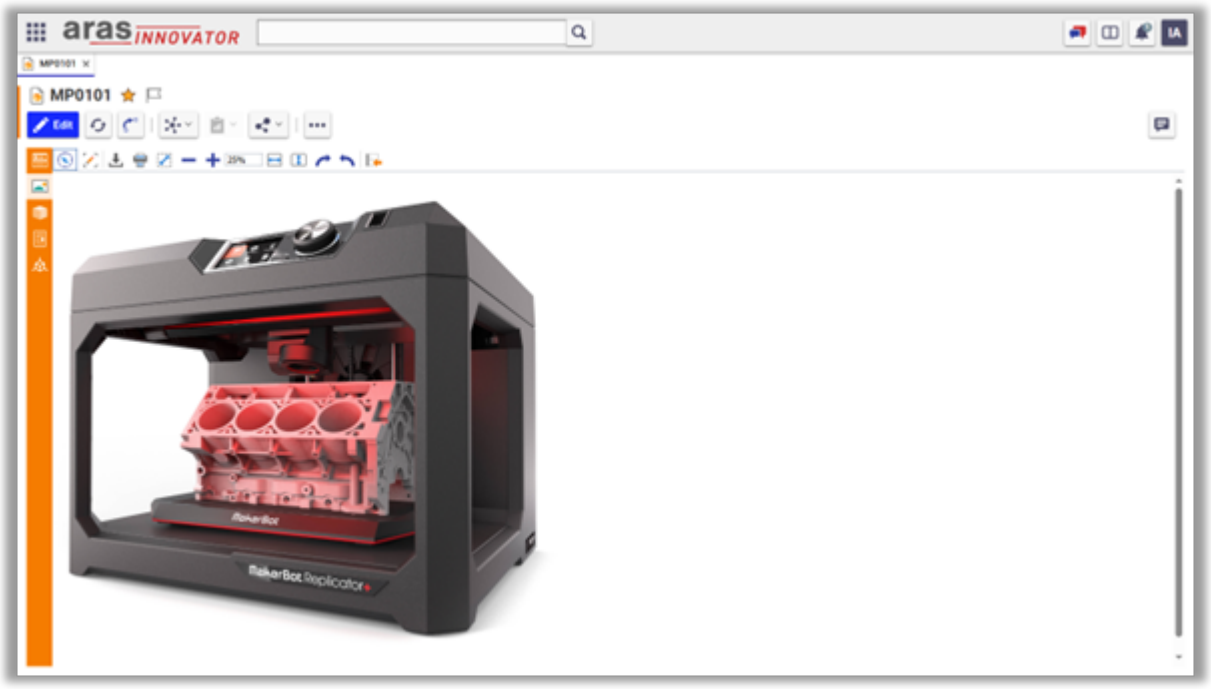
ImageViewer control is used to render SVG and image files associated with a Document item.



- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the control. It can be localized by defined Languages and Locales inside innovator.
- **Height**: ImageViewer control can be rendered in three different modes for the Height.
    - **Height**: Explicit
      In this mode, 'Max Height' should be supplied to limit the height of the control in the page. Control auto grows until it reaches 'Max Height', after that it shows scrollbar.
    - **Height**: Fit to Content
      In this mode, ImageViewer control grows its height without showing any scrollbar. If it doesn't get required height in the page, then page will start showing scrollbar.

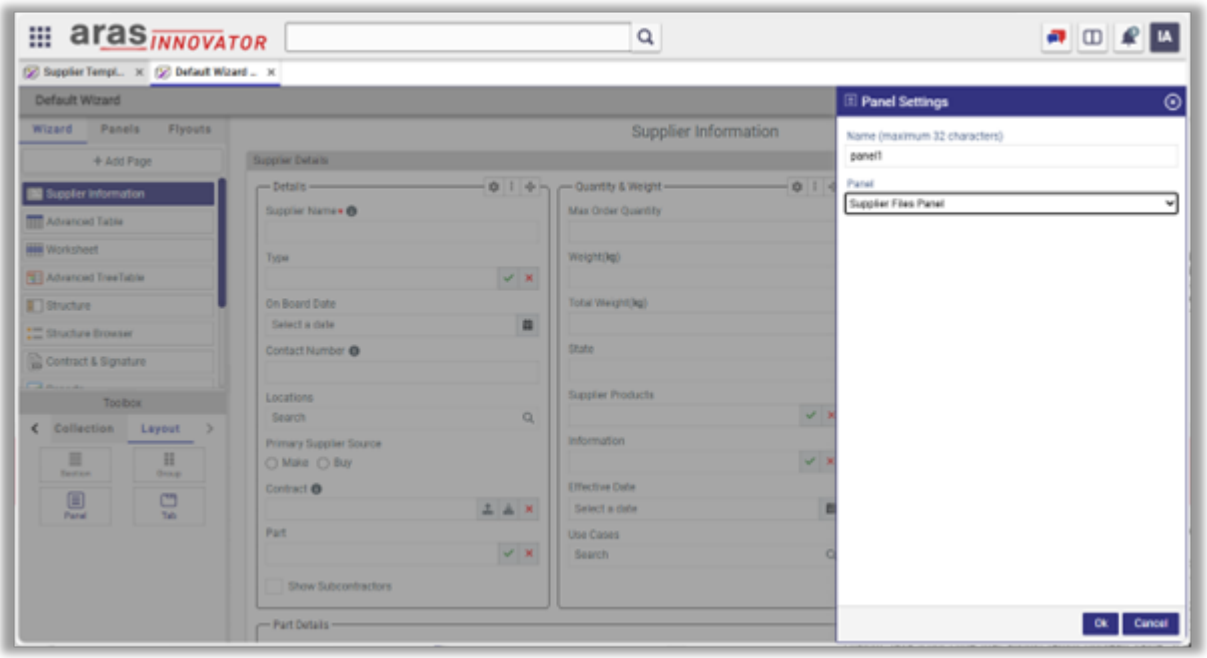
    - **Height**: Auto
      In this mode, control auto grows until it reaches available page height, after that it shows scrollbar for the body. This mode can be set on the ImageViewer control only when the page that contains ImageViewer control has only simple form fields apart from this ImageViewer control.
- **Show Label**: It controls whether Label can be shown at the top of the ImageViewer control.
- **Access**: It allows you to define the roles which can access control. You can configure Roles (in the form group identities) to whom this control should be shown.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.
- **Font**: It allows you to set the required font and size for the control.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **OnLoad**: OnLoad event allows you to write custom logic that will be executed after the data in the control is loaded. It allows you to write a custom script to find and assign Document id to the ImageViewer control. Please refer to the Template Customization section for more details.

Runtime View:



Panel Control

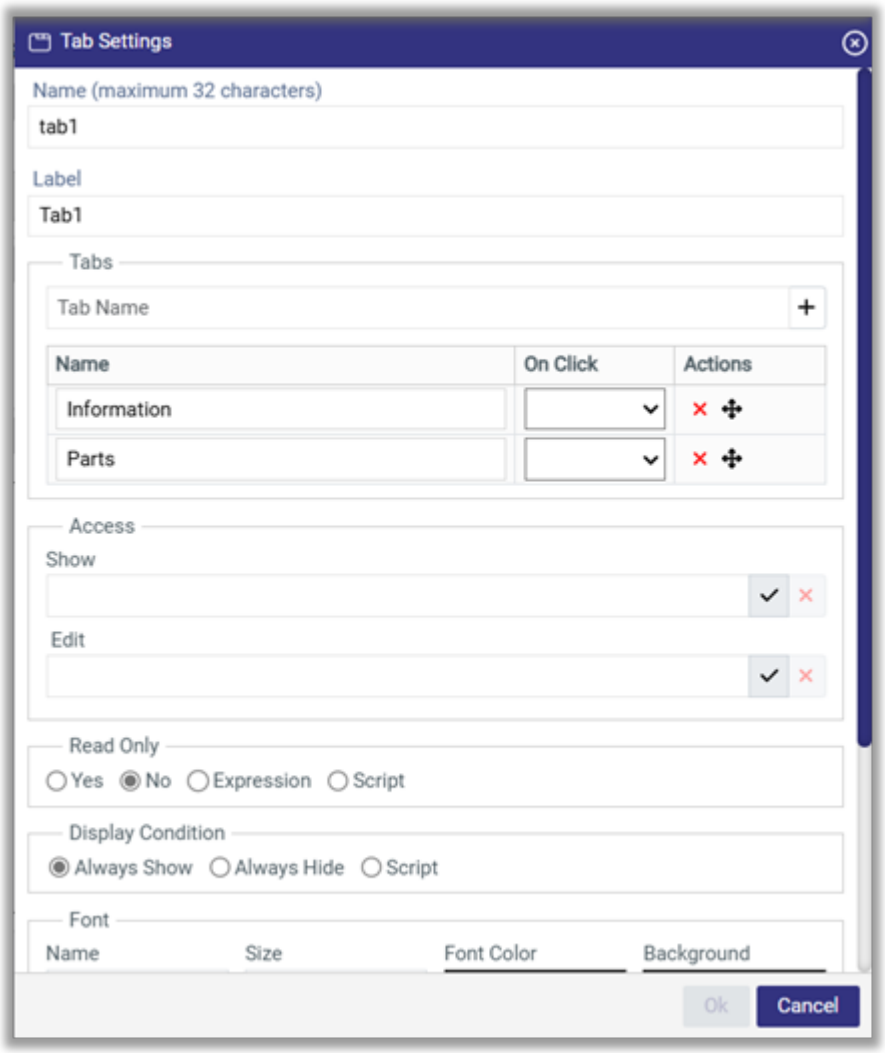
Panel control is the layout control that can help you to create a complex layout on the page. Panel control allows you to use panel defined in the template inside the page layout. Panel control can be placed inside group control just like any other control.



- **Name**: It is used to uniquely identify control inside the template.
- **Panel**: It shows a list of defined panels in the template. You can select one of the panels that can be referenced by this control.

Tab Control

Tab control allows you to create multiple tabs which are bound to panels in the template. Tab control inherits ‘Associated Type’ set on container controls like Group or Section. If the ‘Associated Type’ on the container is set as ‘Context Type’, it can render data related to the ItemType associated with template. If the ‘Associated Type’ is set as ‘Type’, based on the ItemType selected, Tab control can render the data related to that selected ItemType. In order to bind a panel with a tab, panel must be created with same ‘Associated Type’ as the Tab control. At runtime, while rendering the Tab control, it must have itemId available in the context, otherwise Tab control will be hidden. Tab control context itemId can be set programmatically by handling event from other collection control like Table row selection.

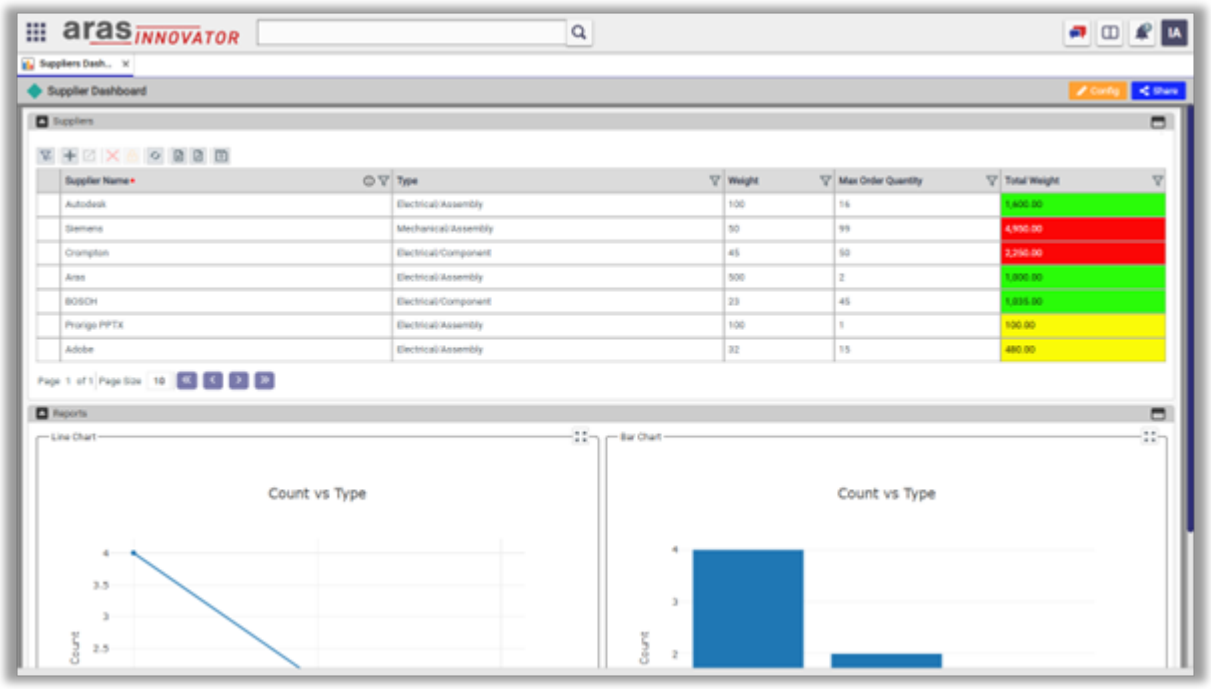


- **Name**: It is used to uniquely identify control inside the template.
- **Label**: It is used to show title on the control. It can be localized by defined Languages and Locales inside innovator.
- **Tabs**: Tab can be created by entering the tab name. Each tab should be associated with a panel by selecting the existing panel from the template under ‘On Click’ column.
- **Access**: It allows you to define the roles which can access control. You can configure Roles (in the form group identities) to whom this control should be shown.
- **Read Only**: Based on the option selected for Read Only, control can become read-only even if you can edit the properties used inside the page with lock. You can select one of the options like ‘Yes’, ‘No’, ‘Expression’, or ‘Script’. Expression allows you to define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally make the control read-only if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is true, the control will be read-only.
- **Display Condition**: Based on the option selected for Display Condition, the control will be shown or hidden in the wizard at runtime. You can select one of the options like ‘Always Show’, ‘Always Hide’, ‘Expression’, or ‘Script’. Expression allows you to

define a Boolean expression using properties from the current and previous pages (in case of multi-page application) to conditionally display the control if expression is evaluated to true. Expression will be evaluated based on the cached property values on the client. Script allows you to write custom JavaScript code that should return Boolean value. If returned value is false, the control will be hidden.

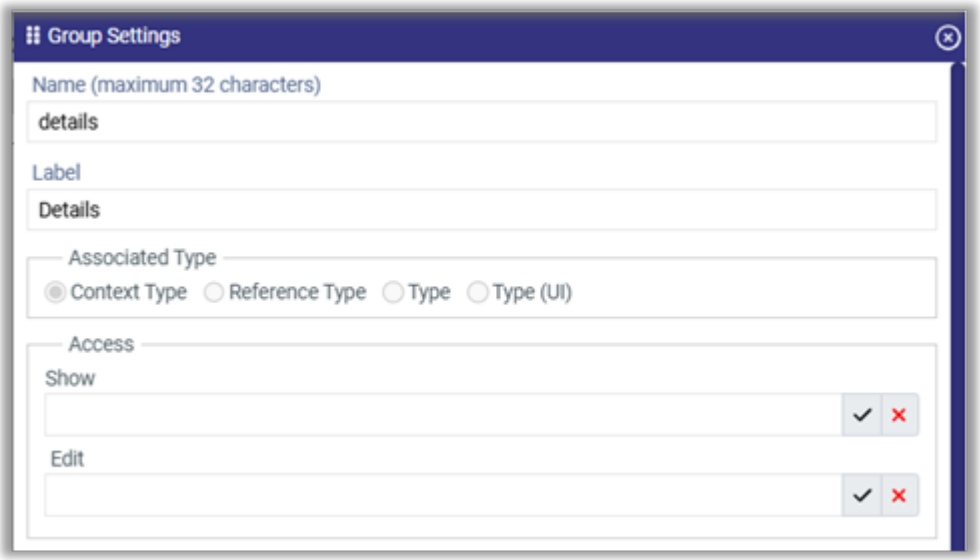
- **Font**: It allows you to set the required font and size for the control.
- **Margin**: It allows you to set the required top, right, bottom, left margins with respect to adjacent controls.
- **OnLoad**: OnLoad event allows you to write custom logic that will be executed after the data in the control is loaded. Please refer to the Template Customization section for more details.

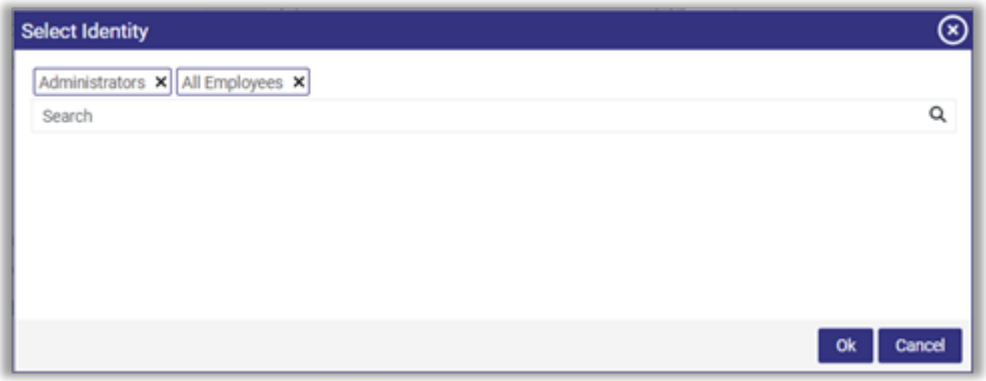
Runtime View:



Access Check

Access setting on each control shows certain portions of the page to users with certain roles. Using this feature, you can build a single template with all the UI elements and define access setting on certain UI controls to hide or show UI based on specific identities. Access setting can be set on a container as well as individual controls. Properties flyout for each control shows following UI to set Show and Edit identities. If these identities are not set, then control will be shown to all users whoever have access to the item.



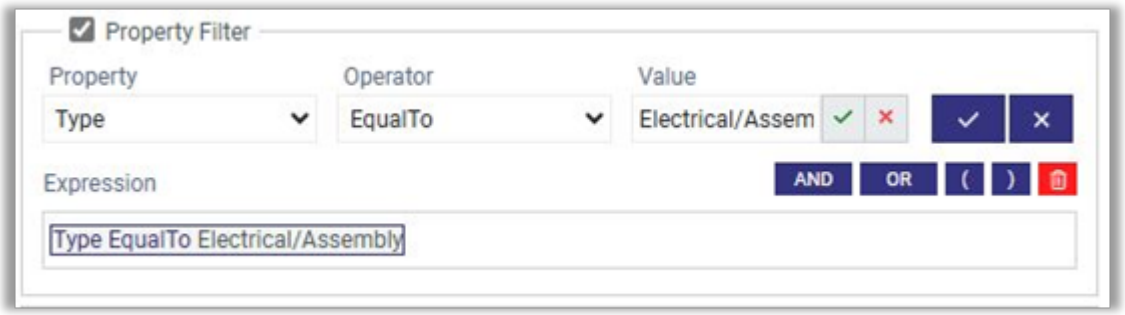


At runtime, Aras ProAppDesigner wizard gets list of accessible controls by checking if the you are a member of any of the identities set on the control. In case, if you are not member, then that particular control will be hidden from the UI. For container controls like Page, Section and Group, in case you

do not have access, then all children will be hidden from UI. Filtering UI elements based on the current user identities takes place on the server, so there is no way you will get to see property values associated with controls.

Expression Builder

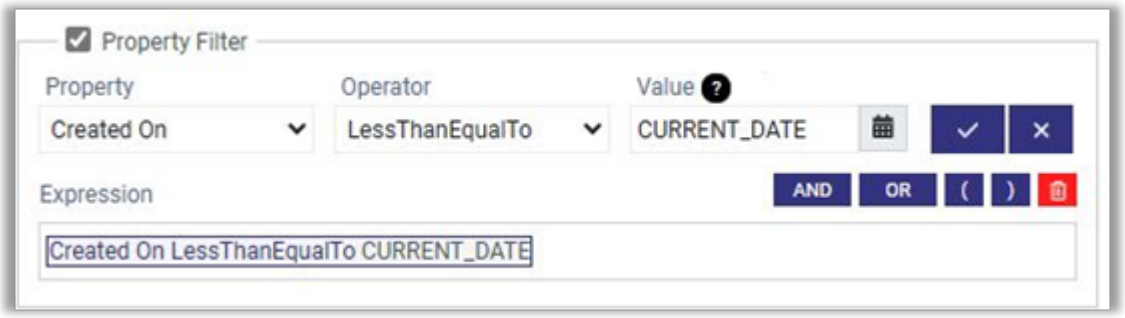
Expression Builder is used at various places in the Aras ProAppDesigner like in creating Display Condition, Default Value, Conditional Formatting, Validation Rules, Property Filter. Using Expression Builder, you can create complex boolean and scalar expressions. Expressions can contain Item properties, operators, values, grouping operator (‘( )’), logical operators (‘AND’, ‘OR’). Various editors are used for showing values based on the selected property type. Existing expression can be updated by selecting individual sub-expression or elements with mouse. You can also navigate expression elements using navigation keys. Selected keyword can be deleted by clicking on the delete icon. If no element is selected, by clicking on delete icon will remove the whole expression.



If no element is selected in the expression, by inserting new property expression or other operators will insert at the beginning of the expression. If any element is selected in the expression, insert operation will insert the new element right after the selected element.

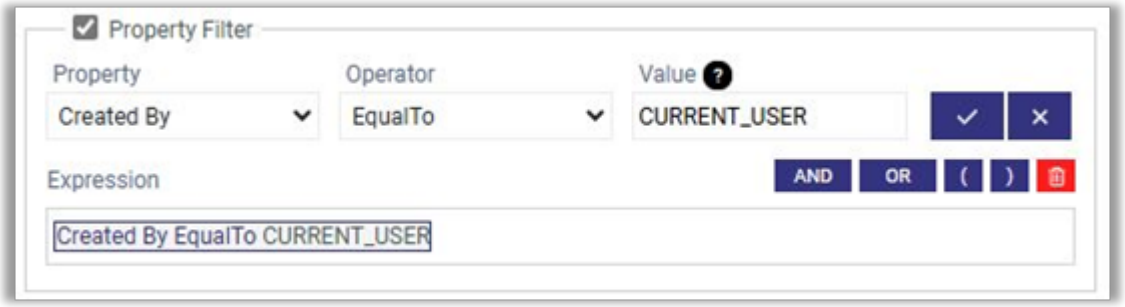
For the date property, Value field can take a specific date or a keyword. In order to build a boolean expression based on current date, value can contain keywords CURRENT_DATE and CURRENT_DATETIME. These keywords will be replaced at runtime before evaluating the expression with actual current date or datetime.

Ex1: **Created On** LessThanEqualTo CURRENT_DATE
Ex2: **Effective Date** LessThanEqualTo CURRENT_DATE + 2
Ex2: **Modified On** LessThanEqualTo CURRENT_DATE – 7



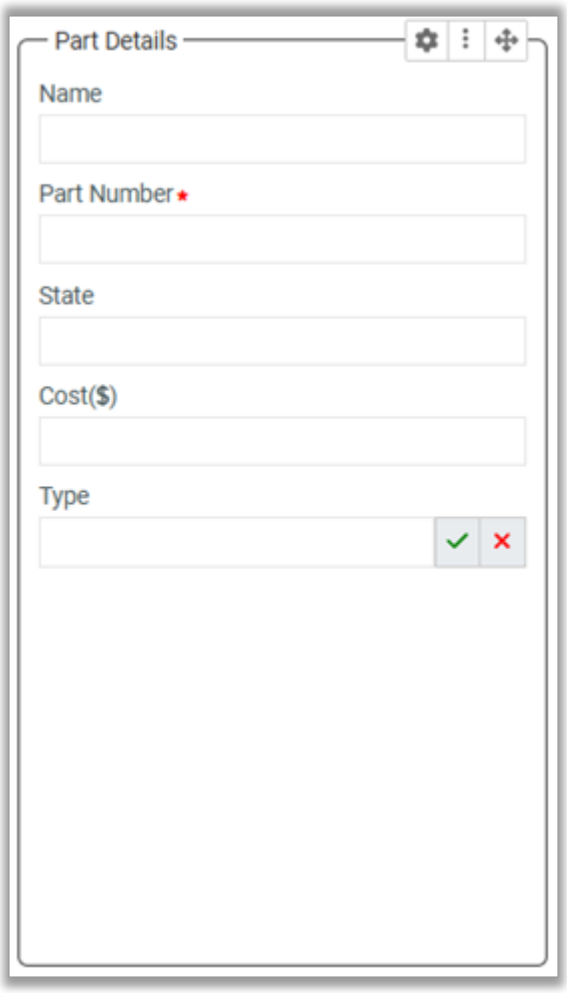
For the user property, Value field can take any text or keyword. In order to build a boolean expression based on current user, value can contain keyword CURRENT_USER. This keyword will be replaced at runtime before evaluating the expression with the logged-in user.

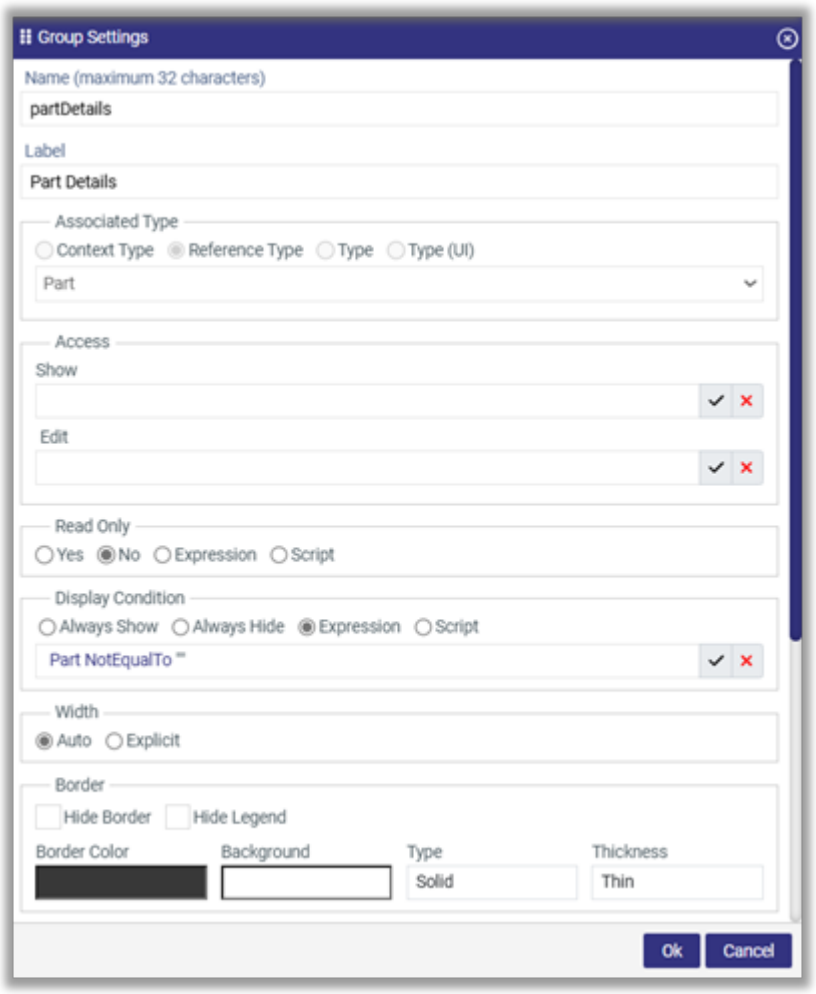
Ex1: **Created By** EqualTo CURRENT_USER
Ex2: **Owned By** NoteEqualTo CURRENT_USER



Associated Item User Interface

- A template can be created based on the associated ItemType properties and relationships.
- The container controls like Section and Group have ‘Associated Type’ setting on Properties flyout, by selecting a value for this setting, related ItemType properties can be shown on the template.
- ‘Associated Type’ setting dropdown shows list of all item properties defined on the ItemType that is associated with the template.
- Once you select value for ‘Associated Type’ setting on Section control, all Group controls placed inside Section control will automatically start showing ‘Associated Type’ setting value from the parent Section control, and this value can’t be changed. Any control placed inside Group control with ‘Associated Type’ setting, can be bound to a property from the associated ItemType, and it can’t be bound to root ItemType properties.

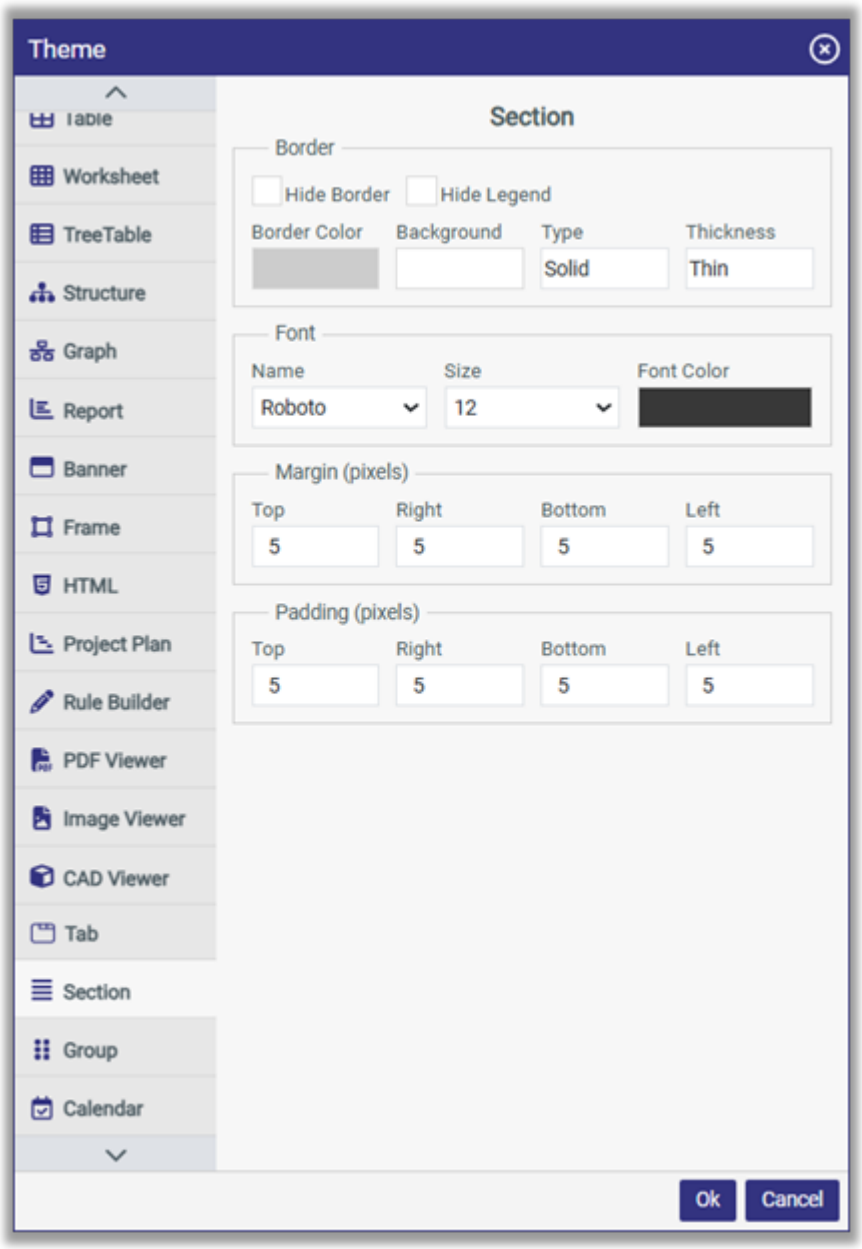




Theme

Theme allows you to style various control types used in the template with common styles globally. It brings uniformity across all controls used in the template. By just changing the style of the specific control type in the Theme, all instance of that control type used in the template will get updated. Style changed through the Theme can be overwritten at individual control level by just changing the settings of that instance in the template. It also has a General section to allow you to style header and footer of the Wizard.

You can change title color for all the section controls in the template by just changing the Theme setting for the section control type.



Control – Property Map

Control Type	ItemType Property	Description
Text Control	String, Text, Integer, Float, Decimal, Date, Sequence, Color	Text control displays properties of datatypes String, Text, Integer, Float, Decimal, Date, Sequence, Color and Classification
List Control	List, Multi Value List, Filter List, Color List	List control displays properties of datatypes List, Filter List and Multi Value List. While displaying list, Aras ProAppDesigner uses same underlining list with options defined in the Innovator.
Checkbox Control	Boolean	Checkbox control displays properties of datatype Boolean from the associated ItemType.
Image Control	Image	Image control displays properties of datatype Image from the associated ItemType.
File Control	File	File control shows properties of datatype File from the associated ItemType.
Document Control	File	Document control selects existing word or pdf document from the Innovator, and that can be embedded in the generated document through Document Generator.
RichText Control	Formatted Text	RichText control displays properties of datatype FormattedText from the associated ItemType. In edit mode, RichText control shows toolbar, using it you can format the text with bold, italic, font, color etc.
Item Control	Item	Item control shows properties of datatype Item from the associated ItemType.
Items Control	Relationship	Items control selects and displays list of items with their keyed-names from the associated ItemType. Selected items will be saved as relationships upon saving the context item.
xClassification Control	xClassification tree	xClassification control associates xClassification tree nodes and xProperties with the item.
ButtonGroup Control		Button control shows a group of buttons or a single button.
Signature Control	Image	Signature control captures your digital signature as image and can be saved to image property of the context item.

Table Control	Relationship & Related ItemType	Table control shows relationship data of the context item.
Worksheet Control	Relationship & Related ItemType	Worksheet control shows relationship data of the context item. Worksheet control gives you experience of working within Excel.
TreeTable Control	Relationship & Related ItemType	TreeTable control shows the structure of items of the same ItemType.
Structure Control	Bound to QueryDefinition	Structure control shows structure data of the context item. Structure control is bound to a Query Definition, that defines which RelationshipTypes, RelatedItemTypes and properties to show in the control. Structure control can also support editing existing structure by inserting or deleting items.
Graph Control	Bound to QueryDefinition	Graph control shows structure data of the context item in graph form. Graph control is bound to a Query Definition, that defines which RelationshipTypes, RelatedItemTypes and properties to show in the control. Graph control can also support editing existing structure by inserting or deleting items.
Report Control	Bound to QueryDefinition	Report control generates reports from available data. Data will be fetched for a report control based on the Query Definition selected for the control.
Banner Control		Banner control shows static html content with pictures to the user. It can be used to show Instructions or Help inside the page.
Frame Control		Frame control shows external content by setting the URL. It can be used to show reports generated through BI tool inside the page.
HTML Control		HTML control shows custom UI in the Aras ProAppDesigner template.
ProjectPlan Control		ProjectPlan control shows project activities and their dependencies.
RuleBuilder Control		RuleBuilder control creates rule template. Rules created based on rule template can be associated with specific ItemType in order to execute on a specific server event.
CAD Viewer Control		CADViewer control renders CAD viewable associated with a CAD Document item.
PDF Viewer Control		PDFViewer control renders PDF documents associated with a Document item.

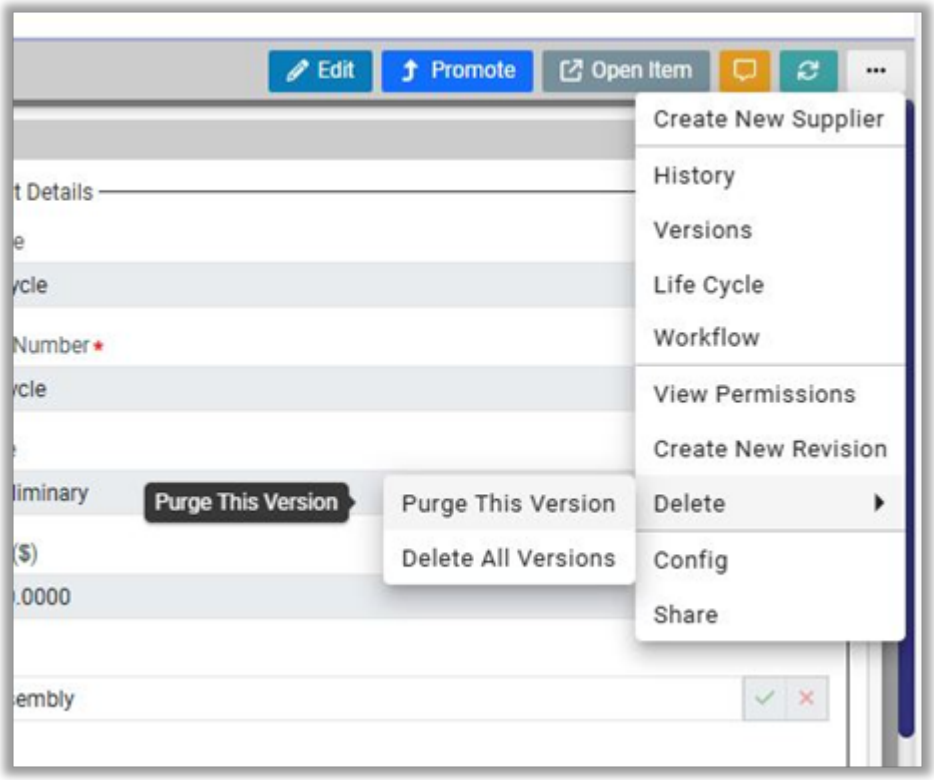
Image Viewer Control		ImageViewer control renders SVG and all image files. It also supports creating markups on the rendered image.
Tab Control		Tab control shows associated pages as tabs.
Calendar Control		Calendar control shows events and project tasks on the resource calendar.
Widget Control		Widget control adds existing widget from the library to the Dashboard and Wizard templates inside a group control.
Panel Control		Panel control displays associated pages as panels.
Section Control		Section control is a layout control to organize group controls placed inside it
Group Control		Group control is a layout control to organize child controls placed inside it.

Wizard

Wizard is a separate single page application (SPA) which shows details of the item at runtime. When you select item from the Innovator search grid, if the item has published Aras ProAppDesigner template, item details with be shown using the template instead of using default Innovator form. When you click on Create action from search grid, the same template will be used to show the page with empty controls.

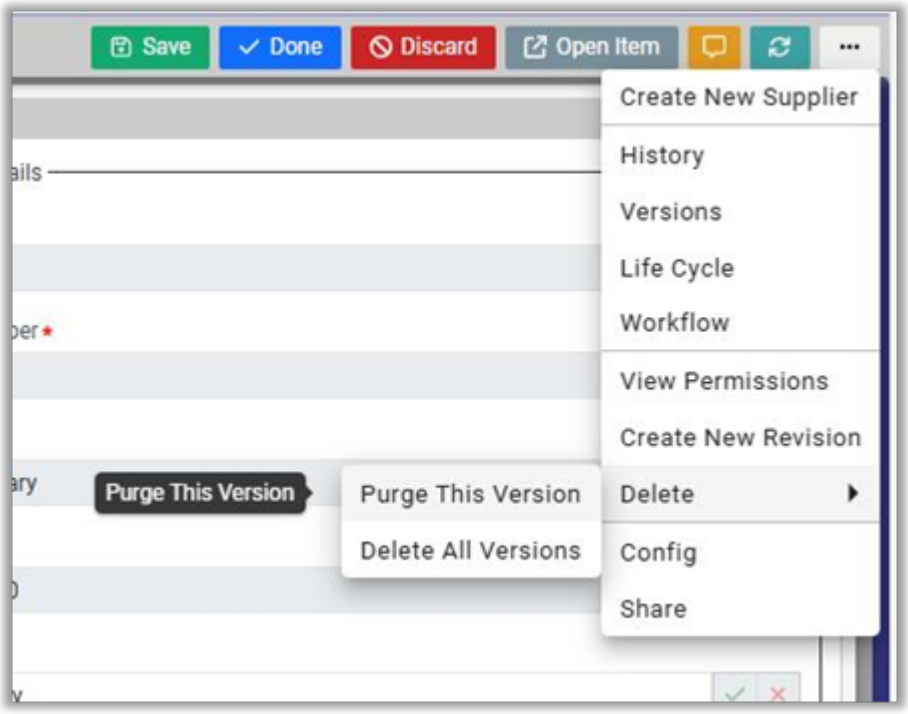
Wizard Actions

Wizard actions are shown in the wizard in the top-right corner when the item is opened. When you open item in read-only mode, you will see actions like Edit, Promote and Open Item.



- **Edit**: Applies lock on the item with exclusive access to make changes on the item.
- **Promote**: Promotes the item to the next state of the lifecycle associated with the item.
- **Open Item**: Shows new Innovator tab with default form that shows item details in Innovator Form view.
- **Discussion Panel**: Shows discussion panel as a flyout with all messages and their replies exchanged in the current item context.
- **Refresh**: Reloads the current item by making server request.
- **Create New Item**: Shows new item page in a different tab.
- **History**: Opens new Innovator tab with default form that shows item details in classic view.
- **Versions**: Shows list of all previous generations of the current item.
- **Life Cycle**: Shows associated life cycle of the current item.
- **Workflow:** Show associated workflow of the current item.
- **View Permissions**: Shows permissions associated with the current item.
- **Delete**: Allows to delete current generation or all generations of the item.
- **Config**: Config action is shown only when admin has configured the template view to allow end user to customize the layout of the wizard of a given ItemType.

- **Share**: Share action is shown only when admin has configured the template view to allow end user to share the layout of the wizard of a give ItemType with other users.
  When you open the item in edit mode, the following actions will be shown:

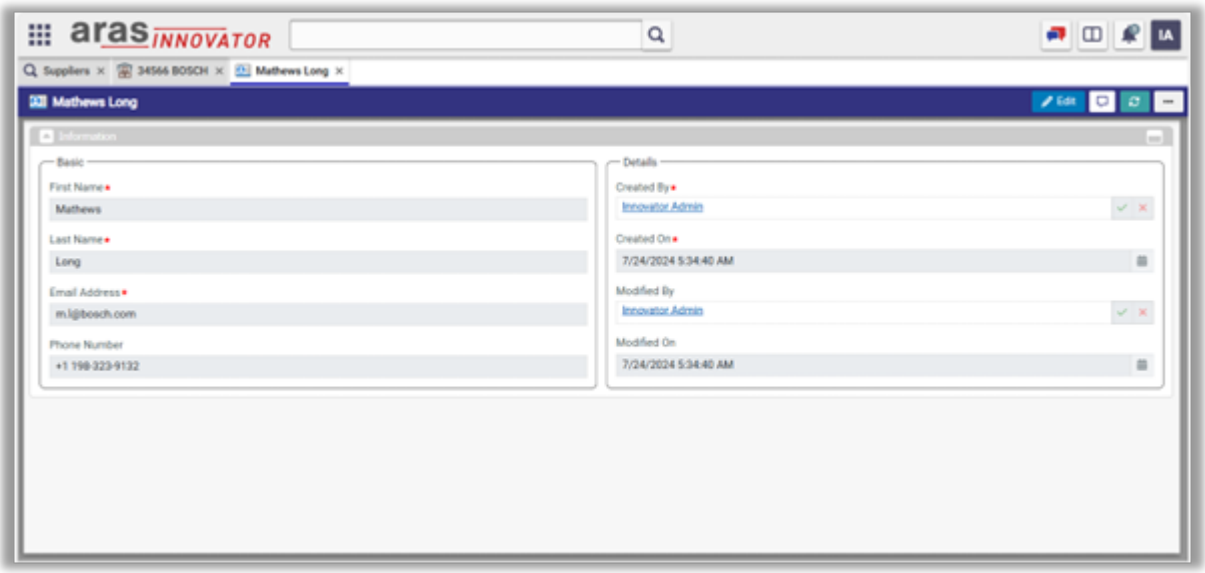


- **Save**: Saves the changes done so far on the wizard. Once changes are saved, you can't discard them using Discard action.
- **Done**: Saves the changes done so far on the item and unlocks the item so that other people can make changes.

- **Discard**: Discards all the changes done so far on the item after the last save and unlocks the item so that other people can make changes.

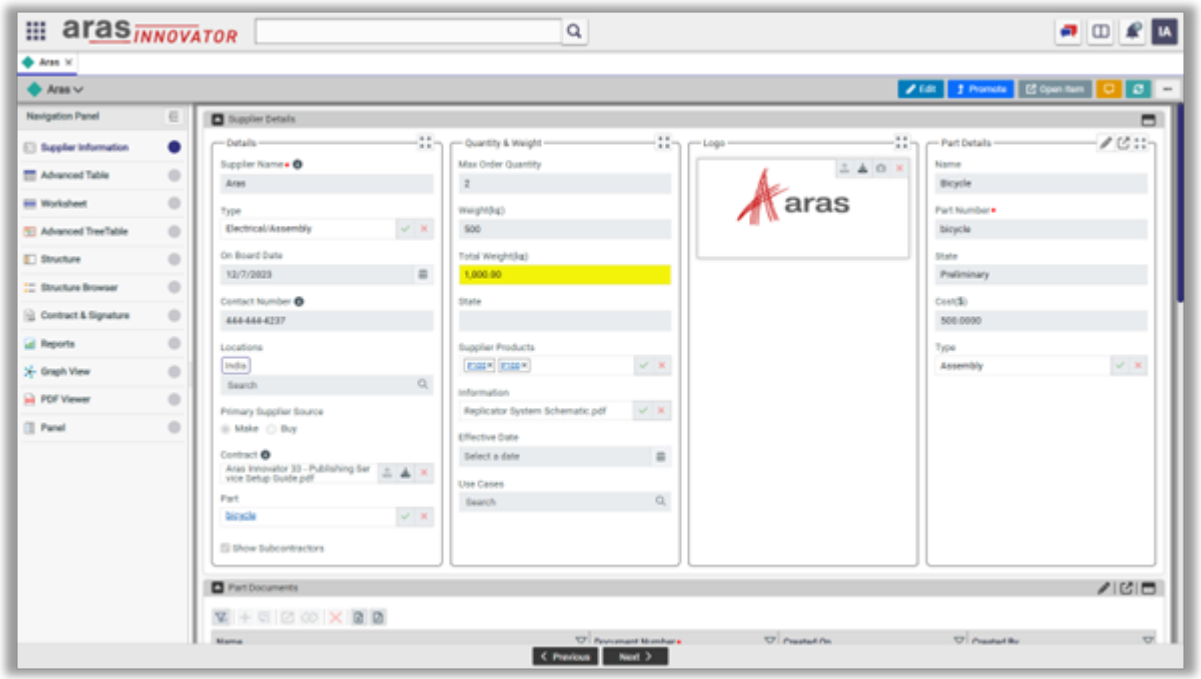
Single Page

The template created in Aras ProAppDesigner can be single page or multi page. If the template has only one page, then it is called a single page application. In case of single page wizard, item keyed-name is used to show the page title.



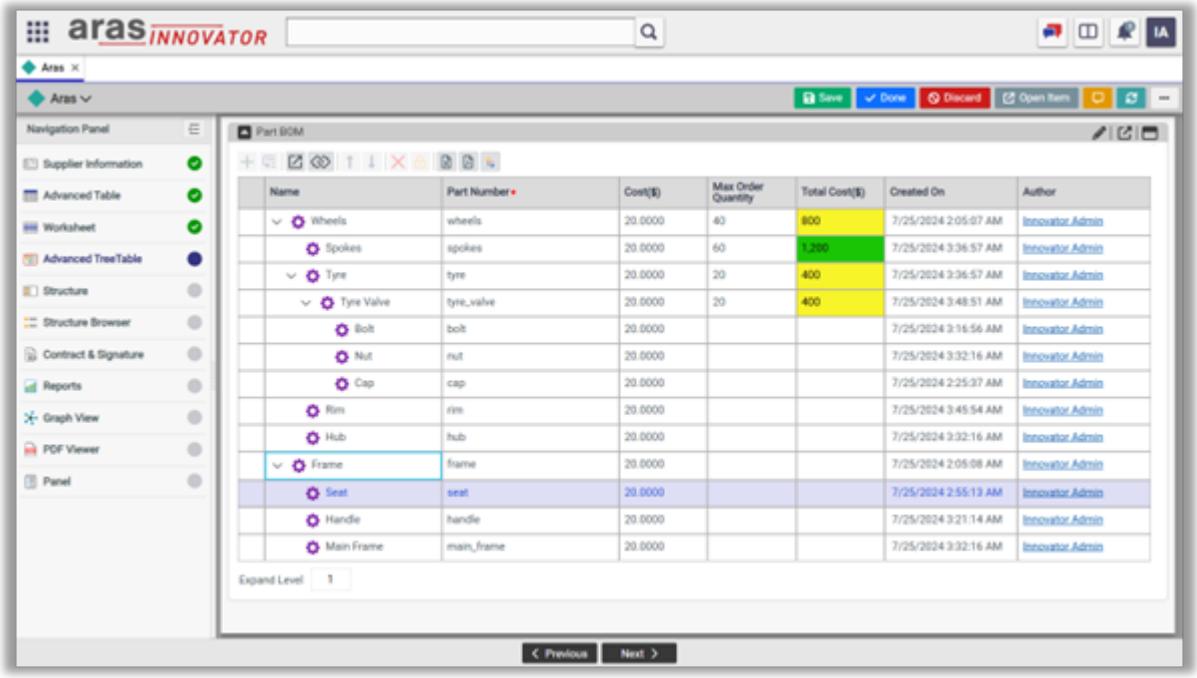
Multi Page

If the template created in Aras ProAppDesigner has more than one page, then it is called multi page application. Multi page application is shown with Navigation panel on the left which displays all the pages of the application. Active page is shown with blue-circle icon, already visited or filled pages are shown with green-circle with tick icon. If the page is not visited, then it will be shown with gray-circle with tick icon. If the page is visited but there are validation errors, then it will be shown with red-circle with tick icon. Along with the Navigation panel, it shows the Navigation Bar at the bottom with buttons Previous and Next. Previous button is disabled when you are on the first page and Next button is disabled when you are on the last page. Sometimes, all the pages may not be shown on the Navigation panel if those pages are defined with Display Condition. Display Condition expression is evaluated every time there is a change in contained property values.



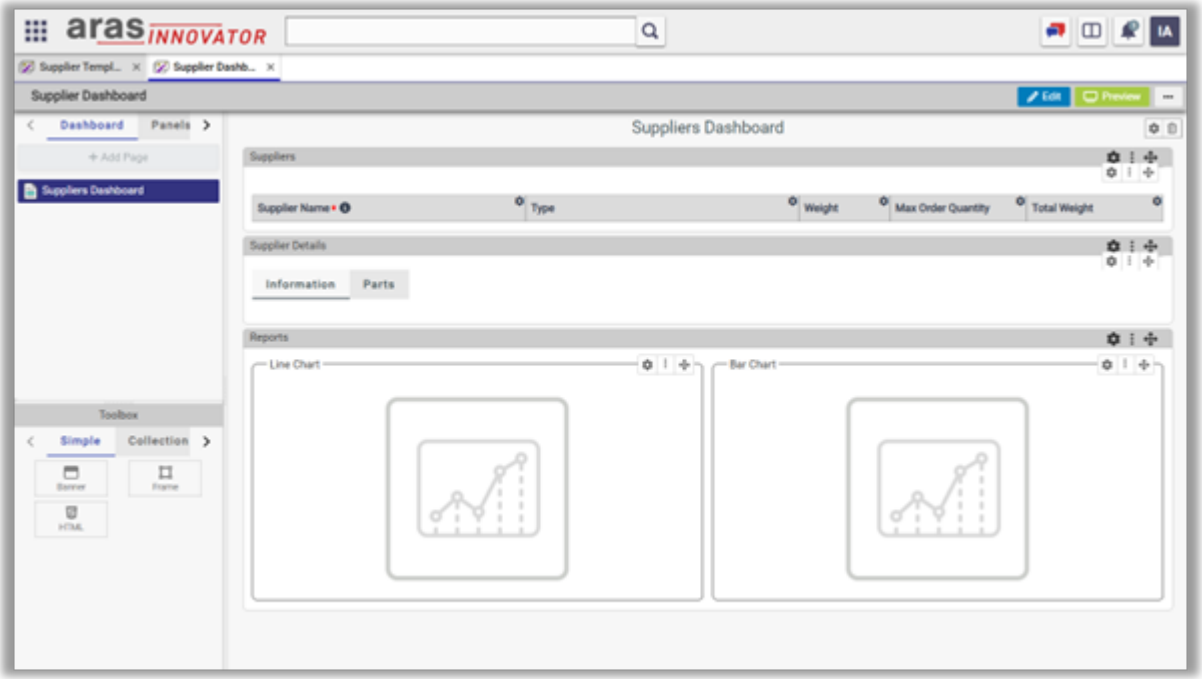
Page Navigation

- You can navigate through the pages by clicking Next and Previous buttons in the bottom Navigation Bar. You can navigate to next page by clicking Next button, as part of moving away from the page, all validation rules on the page are evaluated. If any of the rules are not satisfied, it shows popup dialog with Validations Failed message and icons against each control.
- By hovering the mouse on the icon, specific errors or warning messages are shown based on the defined validation rules on the control. You can move away from the current page by clicking any other page from the Navigation Panel. If the current page has any validation errors, a popup dialog will be shown with options to ignore and move forward or fix the issues before moving forward.
- Every time when you save the item if you have “as_is_validated” property on the ItemType, it will save the status of validations against that property. That property value will be set to true if all pages in the wizard have no validations errors. Even if a single page has validation errors, property value will be set as false.



Dashboards

Dashboards show immediately all the information you are interested in with conditional formatting. Dashboards can have dashboard tables, worksheets, reports, and frame with external content. Dashboard shows dashboard tables and worksheets with pre-applied filters. Dashboard table and worksheet can be shown in edit or read-only mode with conditional formatting and computed columns. Dashboard can be type specific or global, based on the ‘Template Type’ selected at the time of creating the template. Type specific dashboard shows data related to specific ItemType. Global dashboard can show data related to multiple ItemTypes.



Dashboard templates have a single or multi page layout to organize the content. Dashboard template toolbox has only limited set of controls as compared to a regular template. Toolbox has Table, Report, Banner, Frame, Section, and Group controls. Banner, Frame, Section and Group controls have same settings and behave in the same as in the regular template.

Type Dashboard

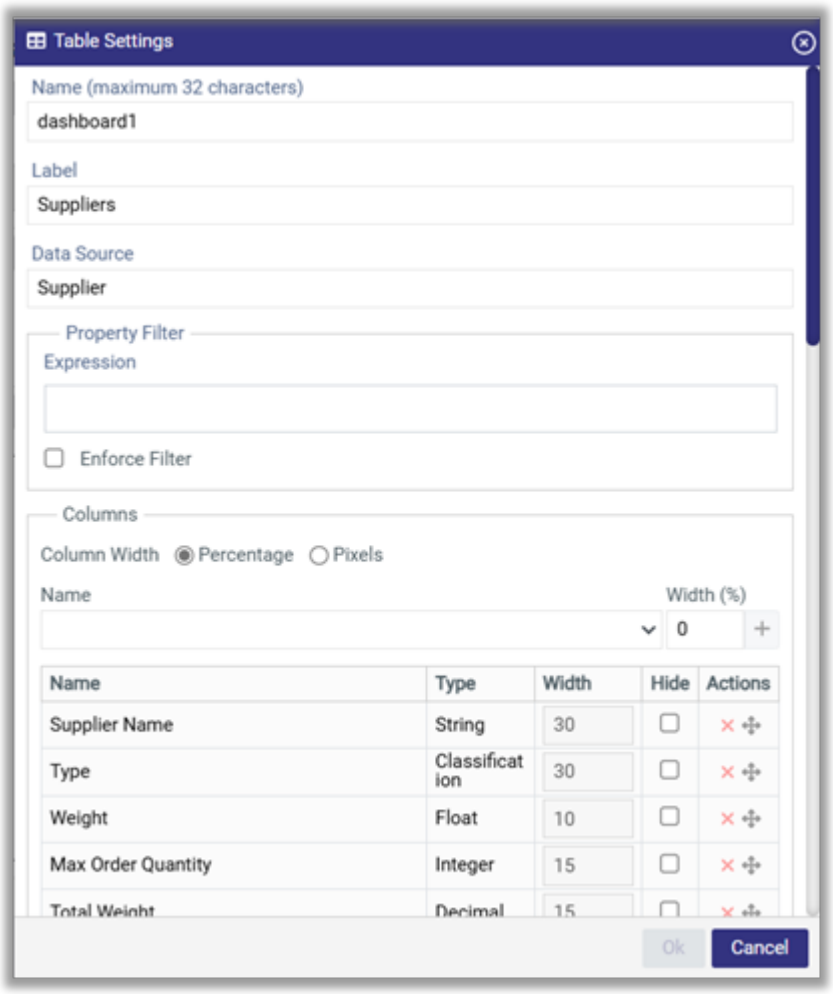
Type dashboard shows all the data related to an ItemType in the form of table and report controls. You can have a table show list of items with pre-applied filter. You can also have reports generated based on that ItemType.

Table Control:

When you add table control to TypeDashboard, its DataSource is set by default with the context ItemType. You can define filter based on the properties to show specific items of interest. As an admin you can enforce the filter, in that case end-user can see the applied filter on the table but can’t change the filter criteria. If enforce filter is not set, end-user can change the filter criteria at the runtime.

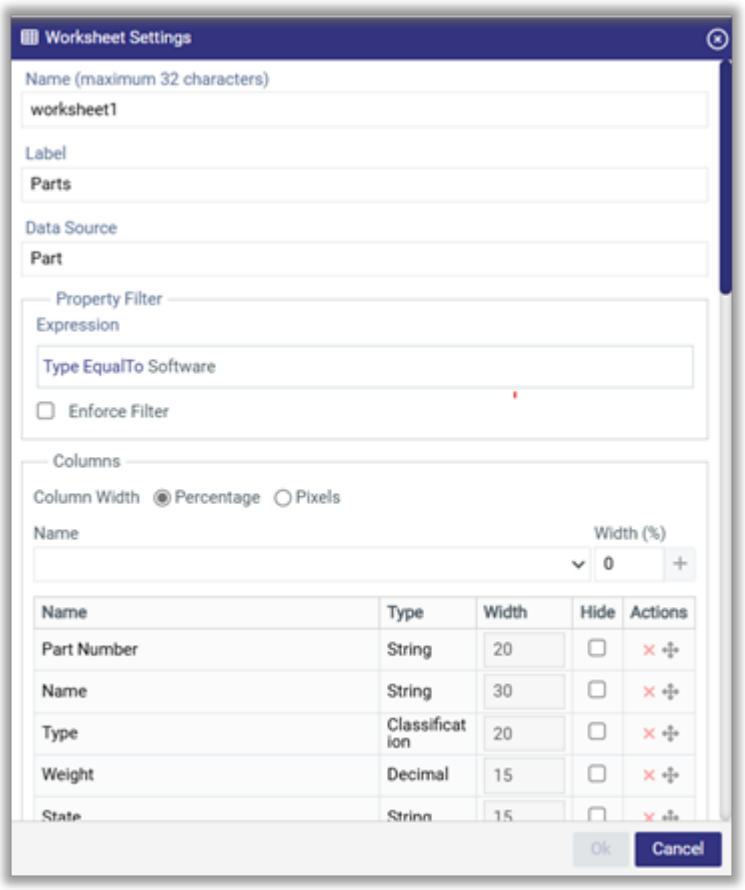
Table control can be shown in the Dashboard in read-only or edit mode by setting the read-only flag. If read-only flag is set to true, end-user can see the data in read-only mode. If the flag is set to false, end-user can edit the item data (inline editing) within the table and save the changes in the bulk without accessing item details page.

You can set Access to a specific role(identity) for entire table or individual columns of table. You can define conditional formatting and computed values on individual columns to highlight certain items.



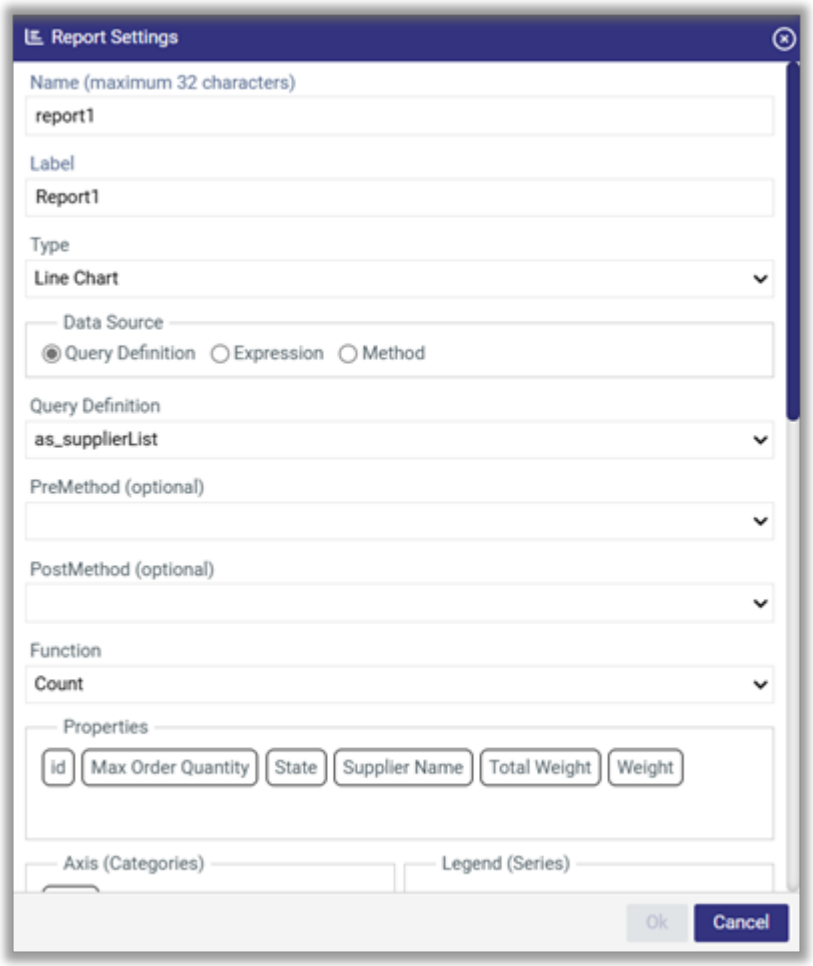
Worksheet Control:

When you add worksheet control to TypeDashboard, its DataSource is set by default with the context ItemType. You can define filter based on the properties to show specific items of interest. You can set Access to a specific role(identity) for entire worksheet or individual columns of worksheet. You can define conditional formatting and computed values on individual columns to highlight certain items. The worksheet can be shown in edit or read-only mode based on the read-only flag.



Report Control:

When you add report control to TypeDashboard, you are supposed to select the QueryDefintion. QueryDefintion dropdown shows you QueryDefintions filtered based on the content ItemType. The rest of the settings on the report control works the same way as defined in the regular template.



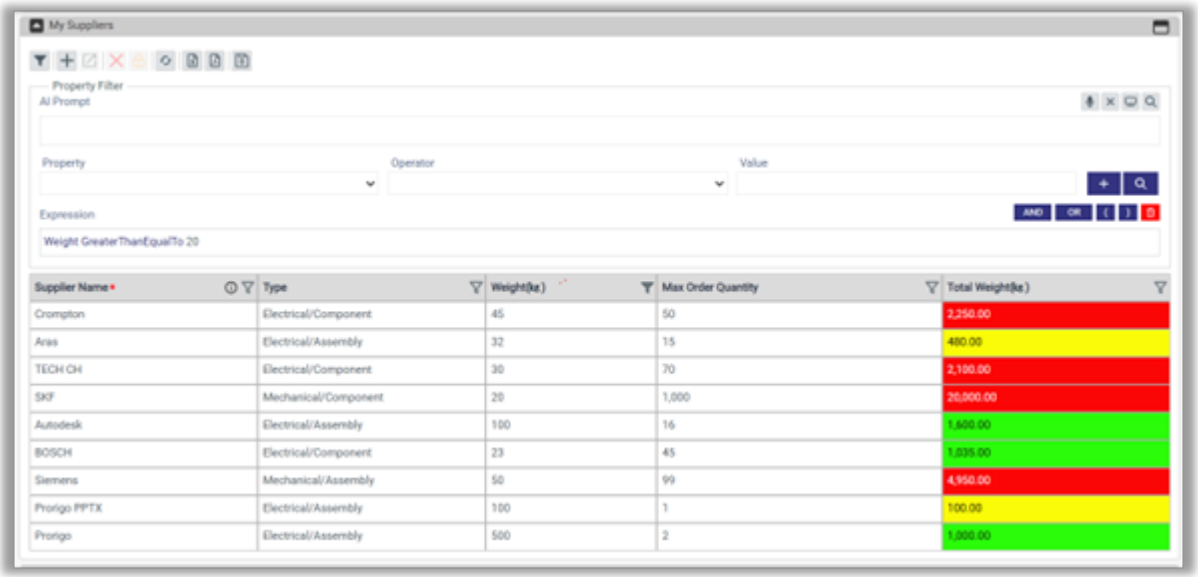
Global Dashboard

Global dashboard shows data related to multiple ItemTypes in the form of table, worksheet, and report controls. You can have a table or worksheet show list of items to a specific ItemType with pre-applied filter. You can also have reports generated based on different ItemTypes.

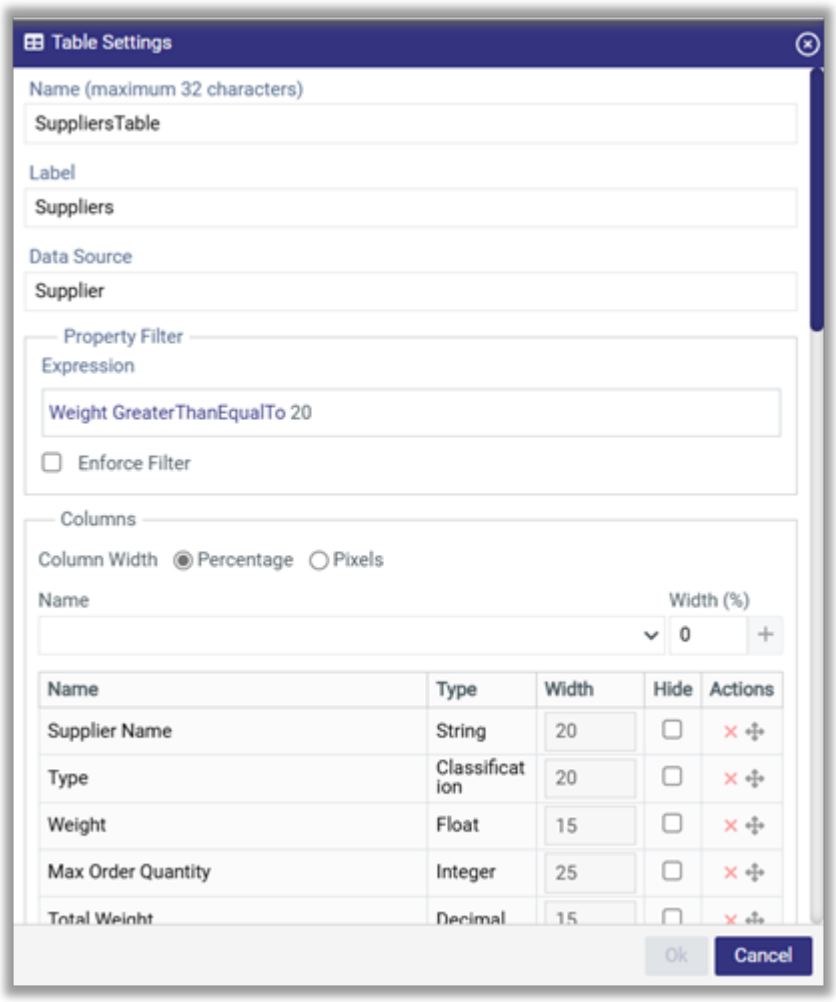
Table Control:

When you add table control to GlobalDashboard, it will ask you to select ItemType as DataSource. After creating table control, DataSource cannot be changed. You can define filter based on the properties to show specific items of interest. As an admin you can enforce the filter, in that case end-user can see the applied filter on the table but can't change the filter criteria. If enforce filter is not set, end-user can change the filter criteria at the runtime.

Table control can be shown in the Dashboard in read-only or edit mode by setting the read-only flag. If read-only flag is set to true, end-user can see the data in read-only mode. If the flag is set to false, end-user can edit the item data (inline editing) within the table and save the changes in the bulk without accessing item details page.

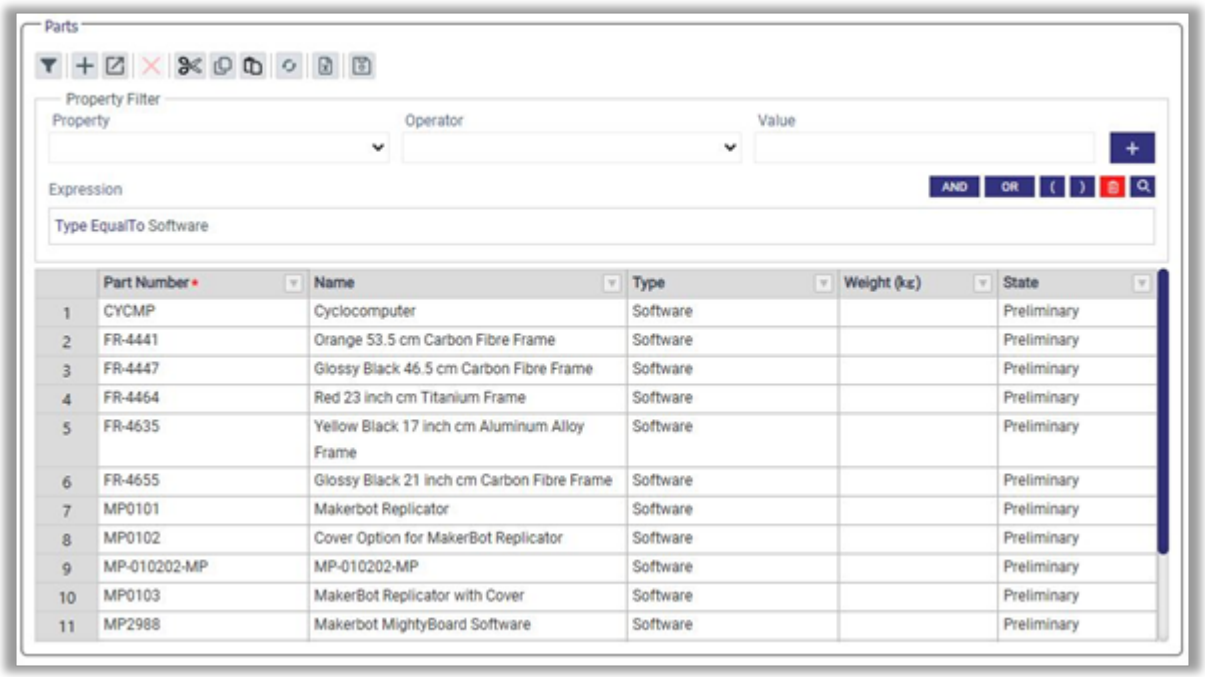


You can set Access to a specific role(identity) for entire table or individual columns of table. You can define conditional formatting and computed values on individual columns to highlight certain items.



Worksheet Control:

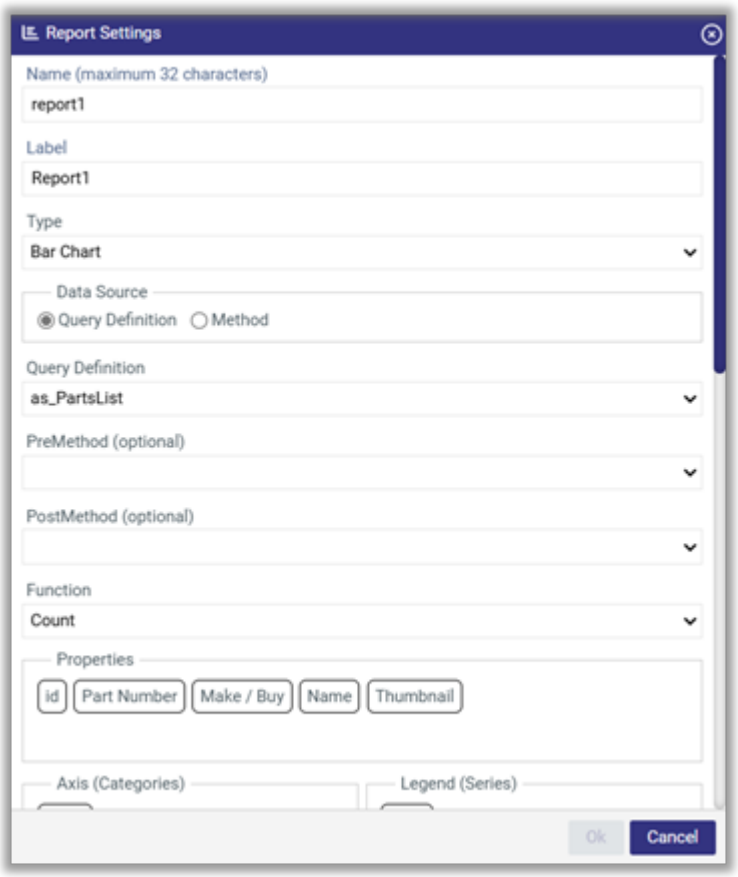
When you add worksheet control to GlobalDashboard, it will ask you to select ItemType as DataSource. After creating worksheet control, DataSource cannot be changed. You can define filter based on the properties to show specific items of interest. As an admin you can enforce the filter, in that case end-user can see the applied filter on the table but can't change the filter criteria. If enforce filter is not set, end-user can change the filter criteria at the runtime. Worksheet control can be shown in edit mode to support bulk editing capabilities with inline editing.



	Part Number	Name	Type	Weight (kg)	State
1	CYCMP	Cyclocomputer	Software		Preliminary
2	FR-4441	Orange 53.5 cm Carbon Fibre Frame	Software		Preliminary
3	FR-4447	Glossy Black 46.5 cm Carbon Fibre Frame	Software		Preliminary
4	FR-4464	Red 23 inch cm Titanium Frame	Software		Preliminary
5	FR-4635	Yellow Black 17 inch cm Aluminum Alloy Frame	Software		Preliminary
6	FR-4655	Glossy Black 21 inch cm Carbon Fibre Frame	Software		Preliminary
7	MP0101	Makerbot Replicator	Software		Preliminary
8	MP0102	Cover Option for MakerBot Replicator	Software		Preliminary
9	MP-010202-MP	MP-010202-MP	Software		Preliminary
10	MP0103	MakerBot Replicator with Cover	Software		Preliminary
11	MP2988	Makerbot MightyBoard Software	Software		Preliminary

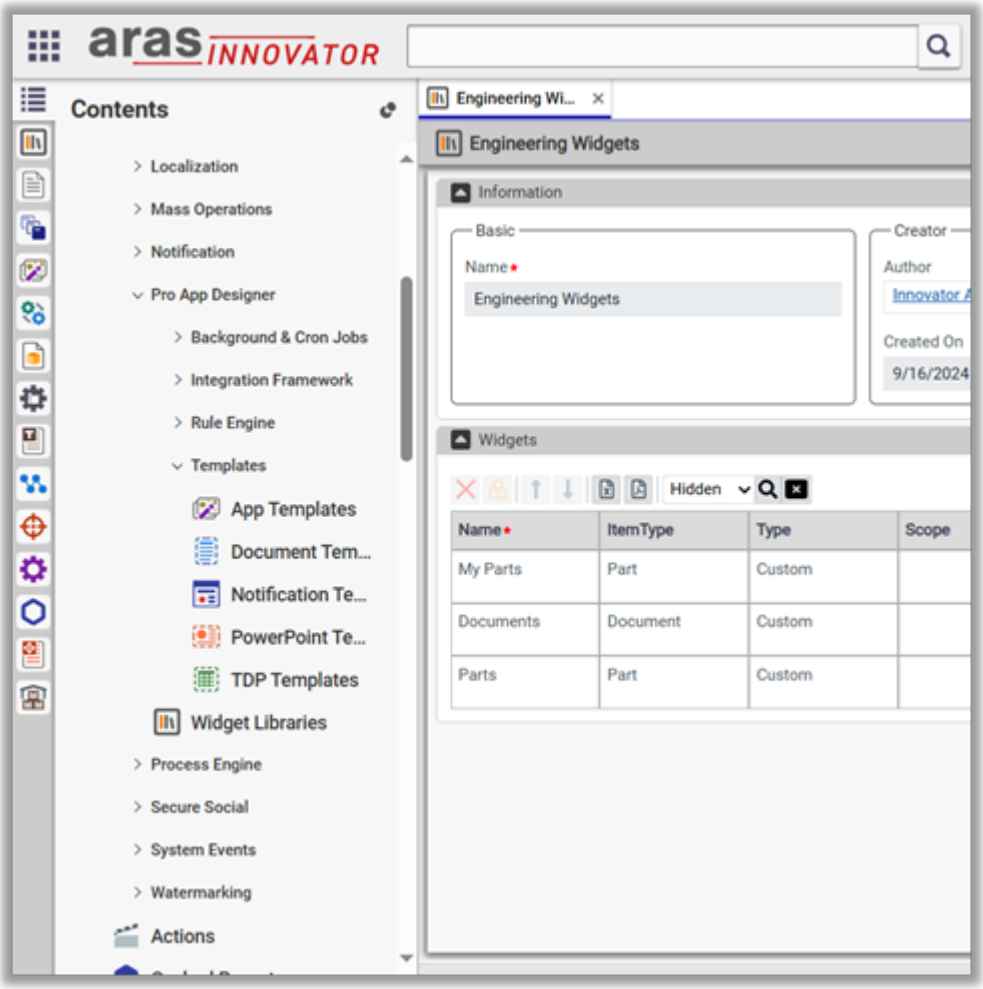
Worksheet Control:

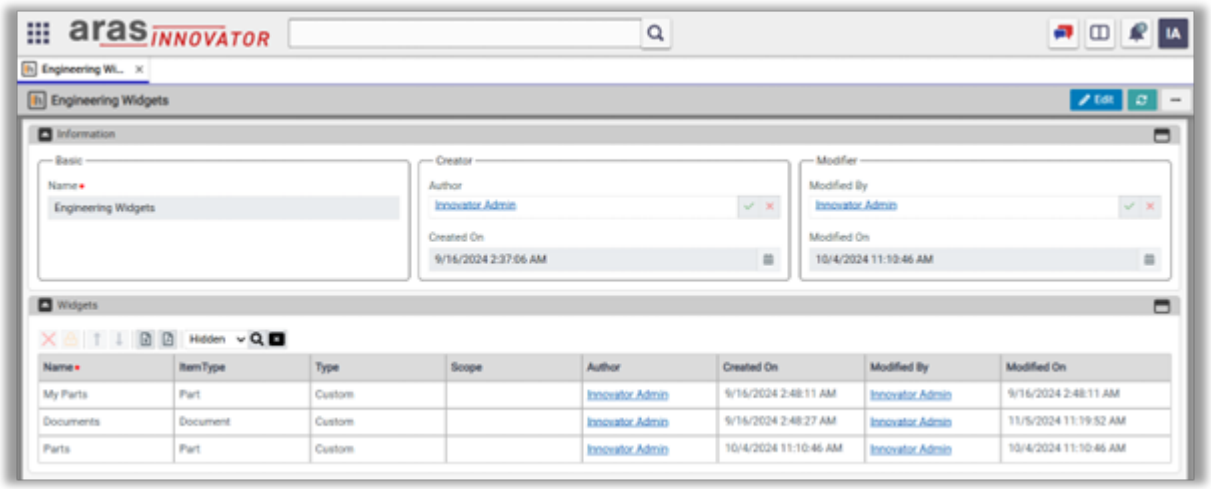
When you add worksheet control to GlobalDashboard, it will ask you to select ItemType as DataSource. After creating worksheet control, DataSource cannot be changed. You can define filter based on the properties to show specific items of interest. As an admin you can enforce the filter, in that case end-user can see the applied filter on the table but can’t change the filter criteria. If enforce filter is not set, end-user can change the filter criteria at the runtime. Worksheet control can be shown in edit mode to support bulk editing capabilities with inline editing.



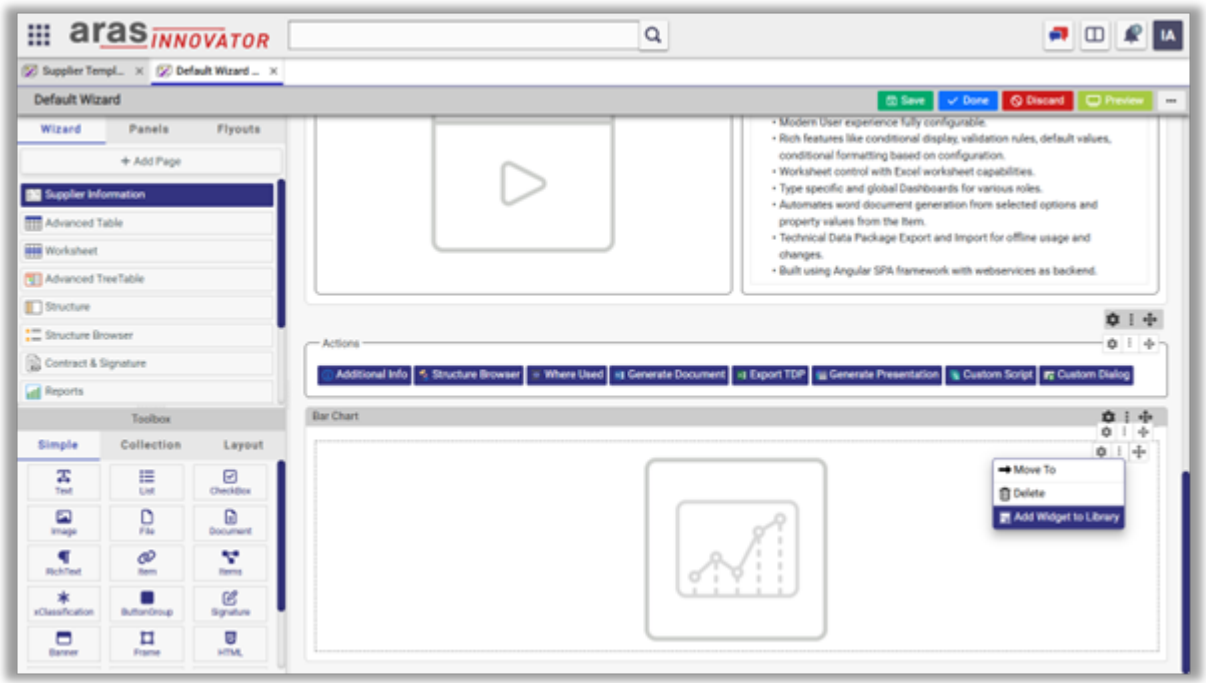
Widget Libraries & Widgets

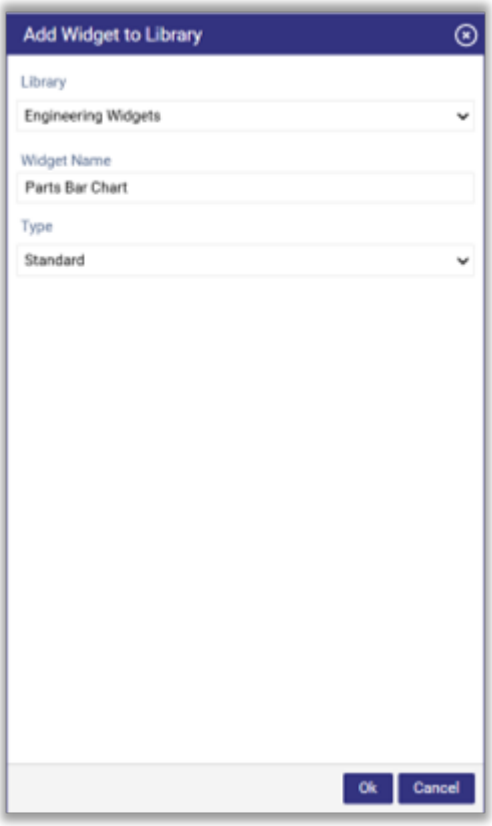
Widgets are predefined UI components, once defined can be used across many templates. Widgets are organized in libraries that are defined by admin. Admin can give access to libraries based on end-user's roles. While creating the template for a specific ItemType or dashboard, admin can select a collection control or the report and add it to the library.



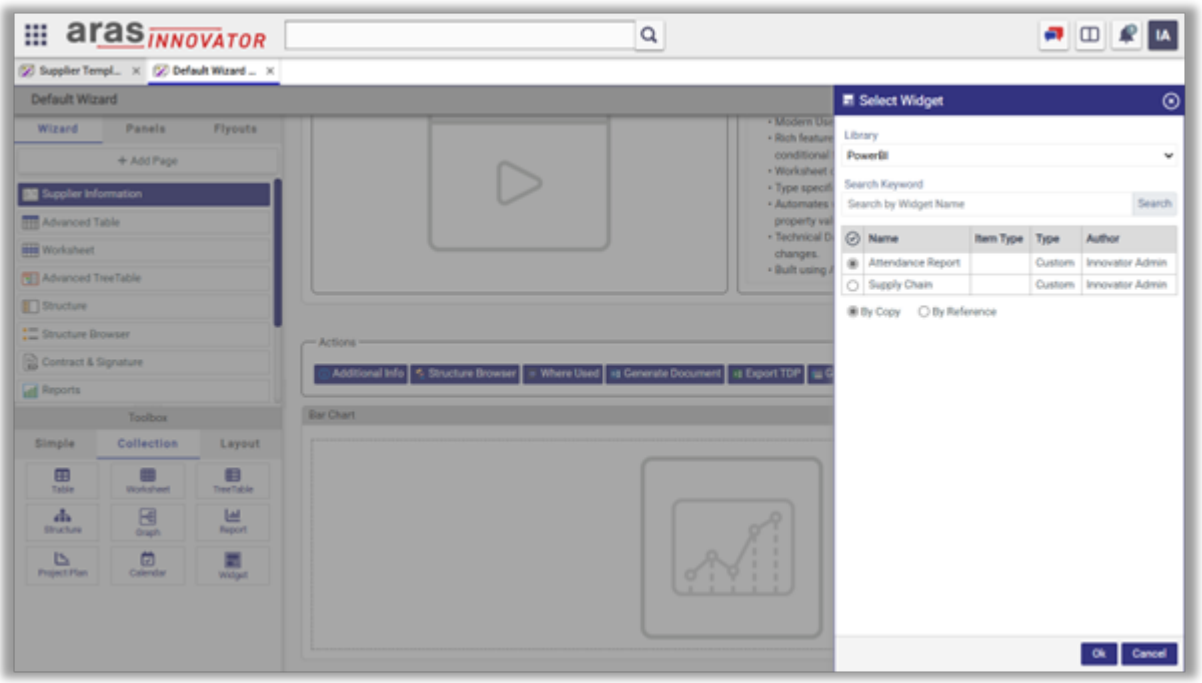


All collection controls and report controls added to the layout, will show the action of ‘Add Widget to Library’ under control dropdown menu. Selecting the action will show the flyout ‘Add Widget to Library’. In the flyout, you can select one of the admins defined libraries, give unique name to the widget, select type as Standard or Custom. Standard widget from the library can be added to the template only by-reference. Custom widget from the library can be added to the template either by-copy or by-reference.





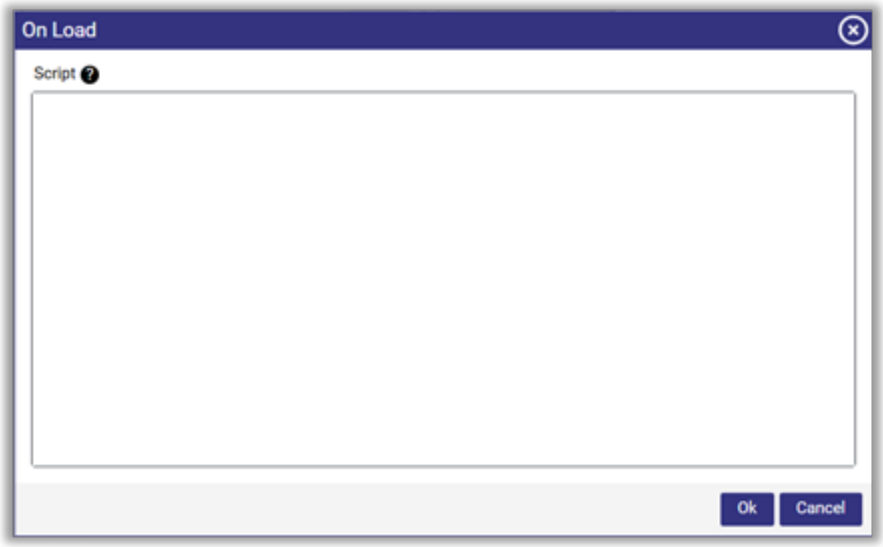
Adding a widget from library to the template can be done using Widget control from the toolbox. Widget can be added to template by-copy or by-reference by selecting appropriate option from ‘Widget Settings’ flyout. Adding widget by-copy, will copy the definition of the Widget to the template. Any subsequent updates done on the widget in the library will not reflect in the template if the widget is added by copy. By adding widget by-reference, the latest widget will always show based on its updates in the library.



Template Customization

Events

Aras ProAppDesigner supports customization through various types of events. These events will be launched by Aras ProAppDesigner Wizard at various instances. As part of creating Aras ProAppDesigner template, admin can register custom script for each of the supported events. Following event types are supported on various controls in TemplateStudio. You can associate custom script with the Event using control settings.



OnClick Event:

OnClick event is supported only on the button control. It will be launched when the button is clicked.

// on button click opening custom flyout

```
appStudioWizard.showDialogFlyout(pageDialogName);
```

OnLoad Event:

OnLoad event will be launched after loading the data for Wizard page or individual collection controls.

```
//Setting list options on a column of the Table control:
const itemId = appStudioWizard.getItemId();
```

*// calling a server method & passing the **itemId** to get the required options. Refer to **10.6 Examples** on how to call server method.*

// building options map

```
let options = {
optionValue: "OptionLabel",
optionValue2: "OptionLabel2"
};
appStudioWizard.setCollectionListOptions("table_name", "column_name", options);
```

OnChange Event:

On Change event will be launched after changing the value of the control. It is supported both on simple controls and collection control columns.

```
// On value change of control1. access its value and set the value of control2
let control1Value = appStudioWizard.getControlValue("control1");
if (control1Value) {
let value = control1Value === "Value1" ? "Discoverable" : "Hidden";
appStudioWizard.setControlValue("control2", value);
}
```

OnBeforeLoad Event:

OnBeforeLoad event will be launched before loading the data in the collection controls inside the page. It is supported only on table, worksheet, treeTable controls. OnBeforeLoad event can be used to set filterExpression on the collection control to load only required dataset.

//Setting filterExpression on Table control

// creating an expression string

```
let expressionString =
"classification EqualTo Item AND name StartsWith AS_";
```

*// Refer to section **10.3.4.9 filterCollectionItems** on how to build your expression string*

// calling filter API for table control

```
appStudioWizard.filterCollectionItems(controlName, expressionString);
```

OnSelect Event:

OnSelect event will be launched when an item in the collection control is selected.

// getting the selected item to access its data

```
const selectedControlItems = appStudioWizard.getControlSelectedItems(
"supplierParts");
if (selectedItems) {
const selectedItem = selectedItems[0];
// accessing data of selected item

const itemId = selectedItem.getItemId();
const relatedId = selectedItem.getRelatedId();
const partWeight = selectedItem.getValue("weight");
}
```

OnSubmit Event:

OnSubmit event will be launched after selecting Submit button in the flyout. It is supported only for the flyouts.

// accessing form values in the flyout & saving them

```
let price = appStudioWizard.getControlValue("price");
let date = appStudioWizard.getControlValue("purchased on");
let managedBy = appStudioWizard.getControlValue("managed_by_id");
```

*// calling a server method to save the data. Refer to **10.6 Examples** on how to call server method.*

```
if (onSuccess) {
appStudioWizard.showSuccessMessage("Item saved successfully");
} else {
appStudioWizard.showErrorMessage("Failed to save the Item.");
}
```

OnInsert Event:

OnInsert event will be launched after Item has been inserted in the collection controls. Event handler will receive inserted item details via eventData, this object will be available in the scope of the event handler code.

```
// Accessing data of inserted item
const insertedItemCount = eventData.length;
for (let i = 0; i < insertedItemCount; i++) {
  // accessing data of selected item

  const relationshipId = eventData[i].getItemId();
const relatedItemId = eventData[i].getRelatedId();
const partWeight = eventData[i].getValue("weight");
   // setting a cell value on inserted item
   eventData[i].setValue("classification", "item");
}
```

OnInsertFilter Event:

OnInsertFilter event will be launched while loading the ItemSelect flyout. Admin can configure a custom script and assign a simple expression string to the eventData, this object will be available in the scope of the event handler code.

```
// creating a custom expression string
 let expressionString = "classification EqualTo Item AND id NotEqualTo
8D58C2D8DC1649279E64CCF5BD2FF7A7";
// Refer to 10.3.4.9 filterCollectionItems on how to build your expression string
```

// assigning expression string to eventData

```
eventData = expressionString;
```

OnSave Event:

OnSave event will be launched after saving the item inside the wizard or section / group controls if they have their own associated items.

```
const itemId = appStudioWizard.getItemId();
const itemType = appStudioWizard.getItemType();
```

*// calling a server method to save custom data. Refer to **10.6 Examples** on how to call server method.*

```
if (onSuccess)
appStudioWizard.showSuccessMessage("Data saved successfully");
else
appStudioWizard.showErrorMessage("Failed to save the custom data.");
```