Aras Innovator 2023 Release

Content Modeling Framework Guide

Document #: *D-008099*

Last Modified: *12/8/2022*

Copyright Information

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810

Phone: 978-806-9400
Fax: 978-794-9826

E-mail: support@aras.com
Website: https://www.aras.com/

Notice of Rights

Notice of Liability

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

Table of Contents

Send Us Your Comments

Aras Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for future revisions.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where and what level of detail?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, indicate the document title, and the chapter, section, and page number (if available).

You can send comments to us in the following ways:

Email:
TechDocs@aras.com
Subject: Aras Product Documentation

Or,

Postal service:
Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Attention: Aras Technical Documentation

If you would like a reply, provide your name, email address, address, and telephone number.

If you have usage issues with the software, visit https://www.aras.com/support/

Document Conventions

The following table highlights the document conventions used in the document:

Convention	Description
Bold	This shows the names of menu items, dialog boxes, dialog box elements, and commands. Example: Click **OK**.
`Code`	Code examples appear in `courier` text. It may represent text you type or data you read.
`Yellow highlight`	Code with yellow highlight is used to draw attention to the code that is being indicated in the content.
`Yellow highlight with red text`	Red color text with yellow highlight is used to indicate the code parameter that needs to be changed or replaced.
Italics	Reference to other documents.
Note:	Notes contain additional useful information.
Warning	Warning contains important information. Pay special attention to information highlighted this way.
Successive menu choices	Successive menu choices may appear with a greater than sign (-->) between the items that you will select consecutively. Example: Navigate to **File** --> **Save** --> **OK**.

1 Content Modeling Framework

1.1 About CMF

The Content Modeling Framework (CMF) is a modeling capability that is used to create a hierarchical data model of independent elements that, when instantiated, are collectively treated as a single entity (CMF Document). CMF also provides a tabular editor for the content of CMF documents, like an Excel spreadsheet.

For an example of CMF in an Aras Innovator solution, refer to the Documents and Views used for FMEA content in the Quality Management System Solution (QMS 11.0R1). Both Design Quality Documents (DQD) and Process Quality Documents (PQD) use CMF.

2 Administration Configurable Options

The following steps walk through the configuration of CMF for a custom Business Object ItemType. For additional detail on configuring properties, please refer to the property definition tables in each section.

2.1 Business Object ItemType

The Business Object ItemType is an ItemType that will be linked to the Content Modeling Framework ItemType. The Business Object ItemType for the following example is called “MyProcessPlan”. This ItemType does not require special configuration for use with CMF.

1. In the Navigation Pane select **Administration → ItemTypes** in the TOC. The following menu appears:

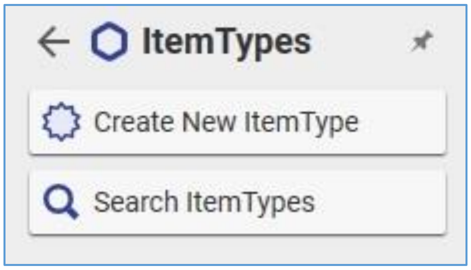


Figure 1.

2. Click **Create New ItemType**. The following screen appears:

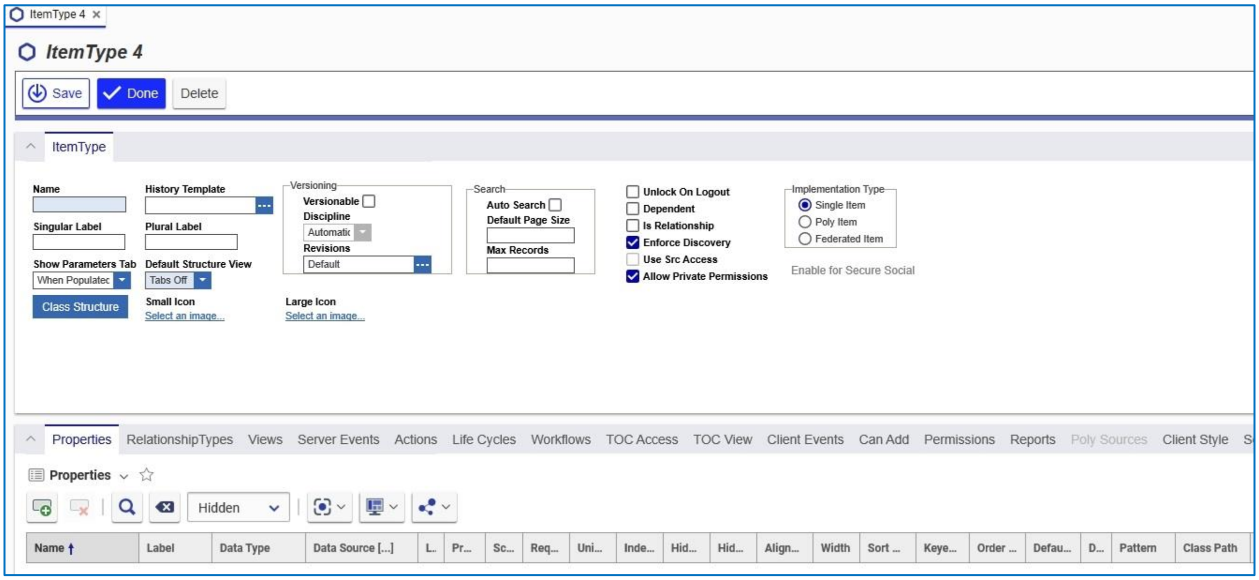
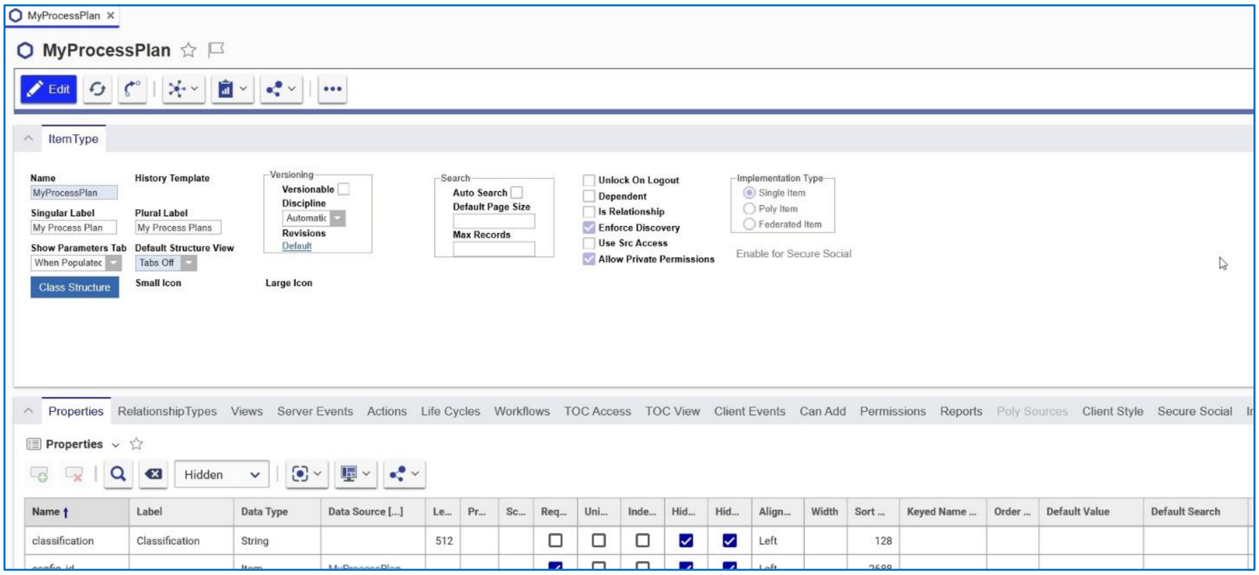


Figure 2.

3. Enter a name and labels.
   a. Name: MyProcessPlan
   b. Singular Label: My Process Plan
   c. Plural Label: My Process Plans

4. In the Properties tab, click to add a new Property.
    a. Name: name
    b. Label: Name
    c. Data Type: String
    d. Length: 32
5. Set access and permissions in the TOC Access tab, the Can Add tab, and the Permissions tab.
6. Click Save and ✓ Done to save and unclaim the MyProcessPlan ItemType. It should look like Figure 3:



Name: MyProcessPlan
History Template
Singular Label: My Process Plan
Plural Label: My Process Plans
Show Parameters Tab: When Populatec
Default Structure View: Tabs Off
Small Icon
Large Icon

Versioning
Versionable □
Discipline: Automatic
Revisions: Default

Search
Auto Search □
Default Page Size
Max Records

□ Unlock On Logout
□ Dependent
□ Is Relationship
☑ Enforce Discovery
□ Use Src Access
☑ Allow Private Permissions

Implementation Type
● Single Item
○ Poly Item
○ Federated Item

Enable for Secure Social

Properties

Hidden

Name ↑	Label	Data Type	Data Source [...]	Le...	Pr...	Sc...	Req...	Uni...	Inde...	Hid...	Hid...	Align...	Width	Sort ...	Keyed Name ...	Order ...	Default Value	Default Search
classification	Classification	String		512			□	□	□	☑	☑	Left		128				
config_id		Item	MyProcessPlan				☑	□	□	☑	☑	Left		2688				

Figure 3.

2.2 CMF ItemType

The CMF ItemType creates a link between the Business Object ItemType and the Content Type, allowing administrators to configure custom views.

Note: All CMF ItemType forms must have an OnFormPopulated event with the core method cmf_ShowContentType.

1. Navigate to **Administration → ItemTypes→Create New ItemType** in the TOC.
2. Enter a name and labels.
    a. Name: MyPQD
    b. Singular Label: My PQD
    c. Plural Label: My PQDs
3. In the Properties tab, add a new Property
    a. Name: name

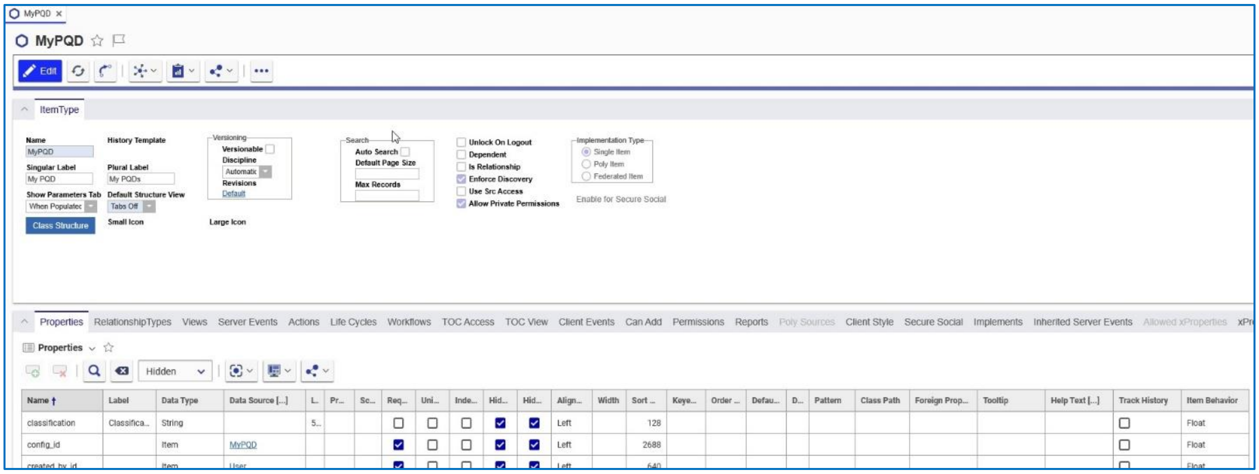
b. Label: Name

c. Data Type: String

d. Length: 32

4. Set access and permissions in the TOC Access tab, the Can Add tab, and the Permissions tab by clicking the Add icon .

5. Click **Save** and **Done**. It should look like Figure 4.



Name: MyPQD

History Template

Singular Label: My PQD

Plural Label: My PQDs

Show Parameters Tab: When Populatec

Default Structure View: Tabs Off

Versioning

Versionable ☐

Discipline: Automatic

Revisions: Default

Search

Auto Search ☐

Default Page Size

Max Records

☐ Unlock On Logout

☐ Dependent

☐ Is Relationship

☑ Enforce Discovery

☐ Use Src Access

☑ Allow Private Permissions

Implementation Type

● Single Item

○ Poly Item

○ Federated Item

Enable for Secure Social

Name ↑	Label	Data Type	Data Source [...]	L..	Pr...	Sc...	Req...	Uni...	Inde...	Hid...	Hid...	Align...	Width	Sort ...	Keye...	Order ...	Defau...	D...	Pattern	Class Path	Foreign Prop...	Tooltip	Help Text [...]	Track History	Item Behavior
classification	Classifica...	String		5...			☐	☐	☐	☑	☑	Left		128										☐	Float
config_id		Item	MyPQD				☑	☐	☐	☑	☑	Left		2688										☐	Float
created_by_id		Item	User				☑	☐	☐	☑	☑	Left		640										☐	Float

Figure 4.

6. Select the **Views** tab and double-click the MyPQD form.

7. Lock the form, go to the Form Event tab, and click **Add Methods** . The Method Search dialog appears:

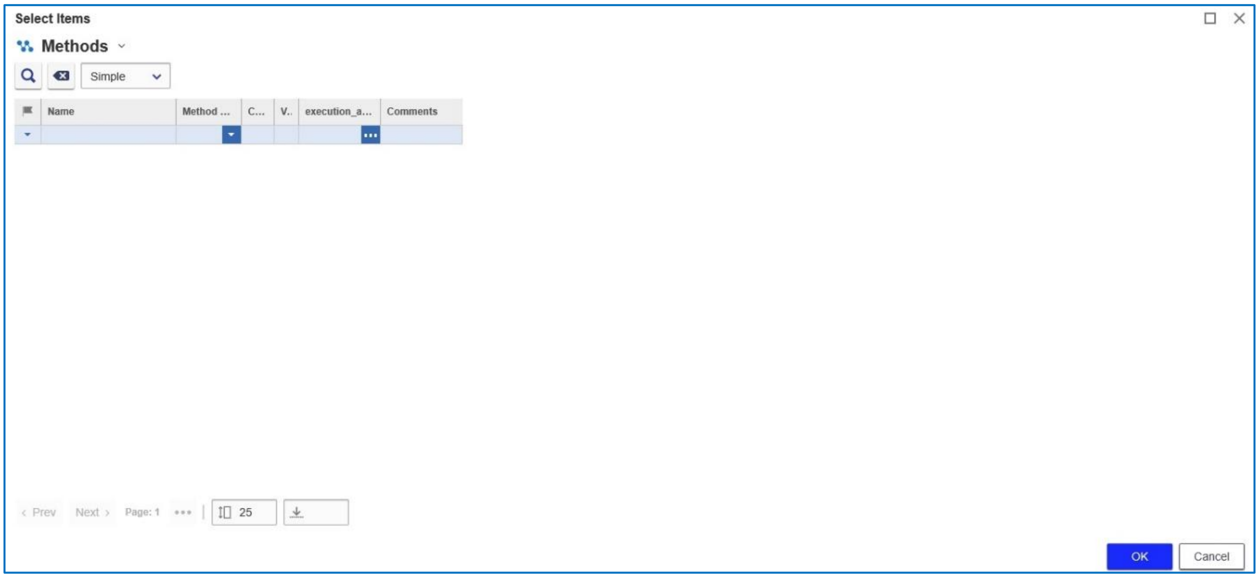
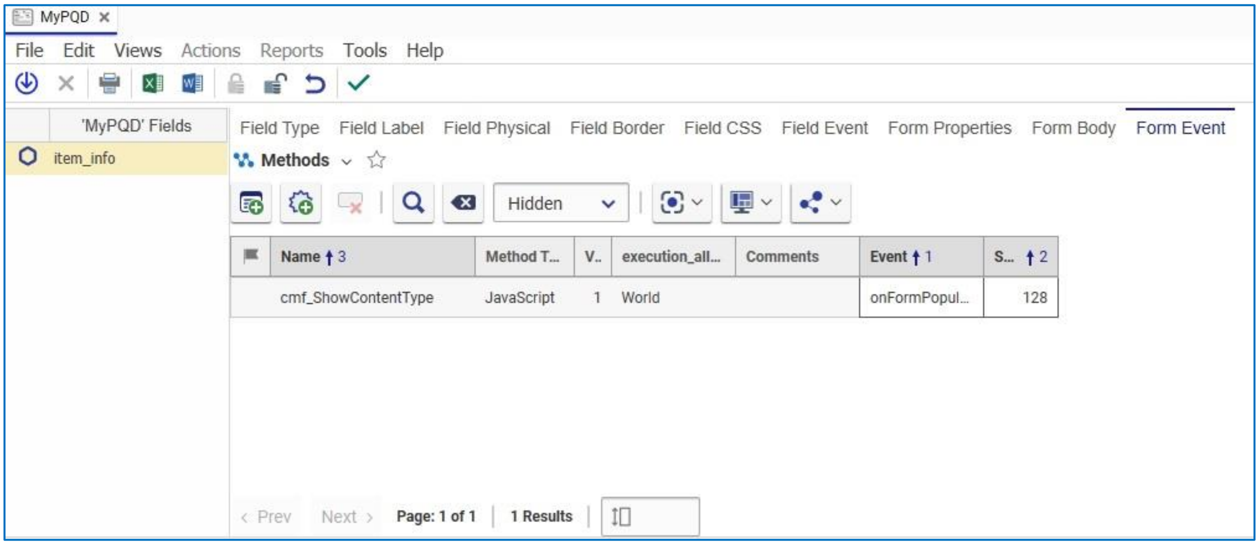


Figure 5.

8. Search for **cmf_ShowContentType**, select it and click **OK**.

9. Double click the blank box in the Event column. Click the Down arrow to select the OnFormPopulated method.

10. Click ⬇ to Save the MyPQD form. Click ✓ Done to unclaim it.



	Name ↑3	Method T...	V..	execution_all...	Comments	Event ↑1	S... ↑2
	cmf_ShowContentType	JavaScript	1	World		onFormPopul...	128

Figure 6.

2.3 CMF RelationshipType

1. Navigate to **Administration → RelationshipTypes** in the TOC. The following menu appears:

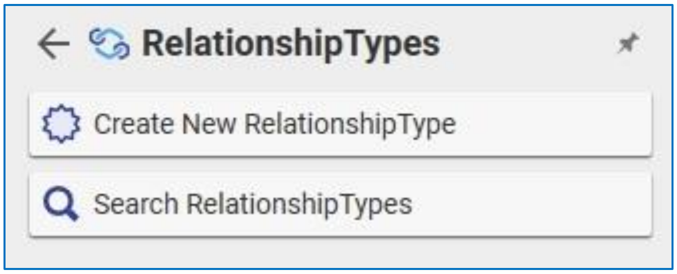
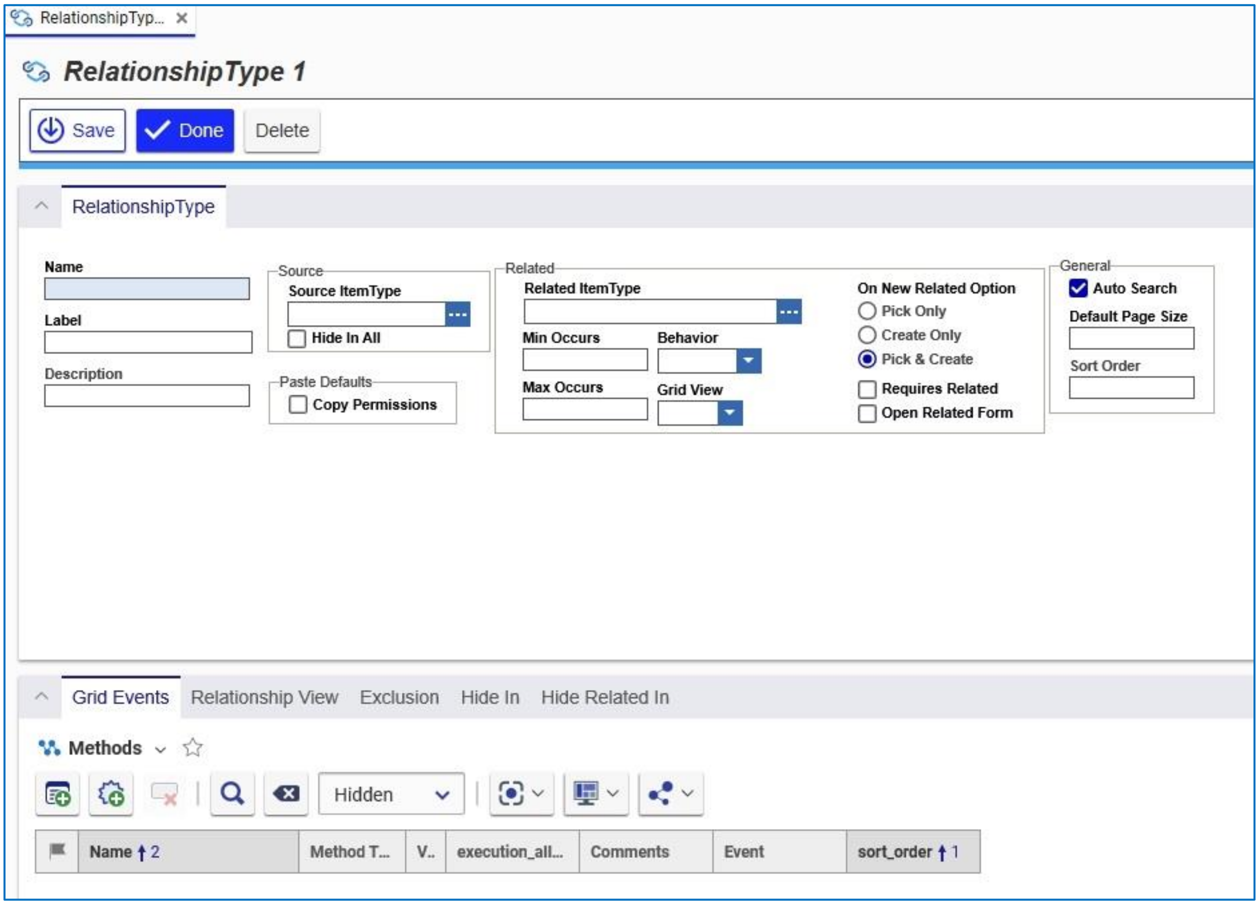


Figure 7.

2. Click **Create New RelationshipType**. The following window appears:



□ Hide In All

□ Copy Permissions

On New Related Option
○ Pick Only
○ Create Only
● Pick & Create

□ Requires Related

□ Open Related Form

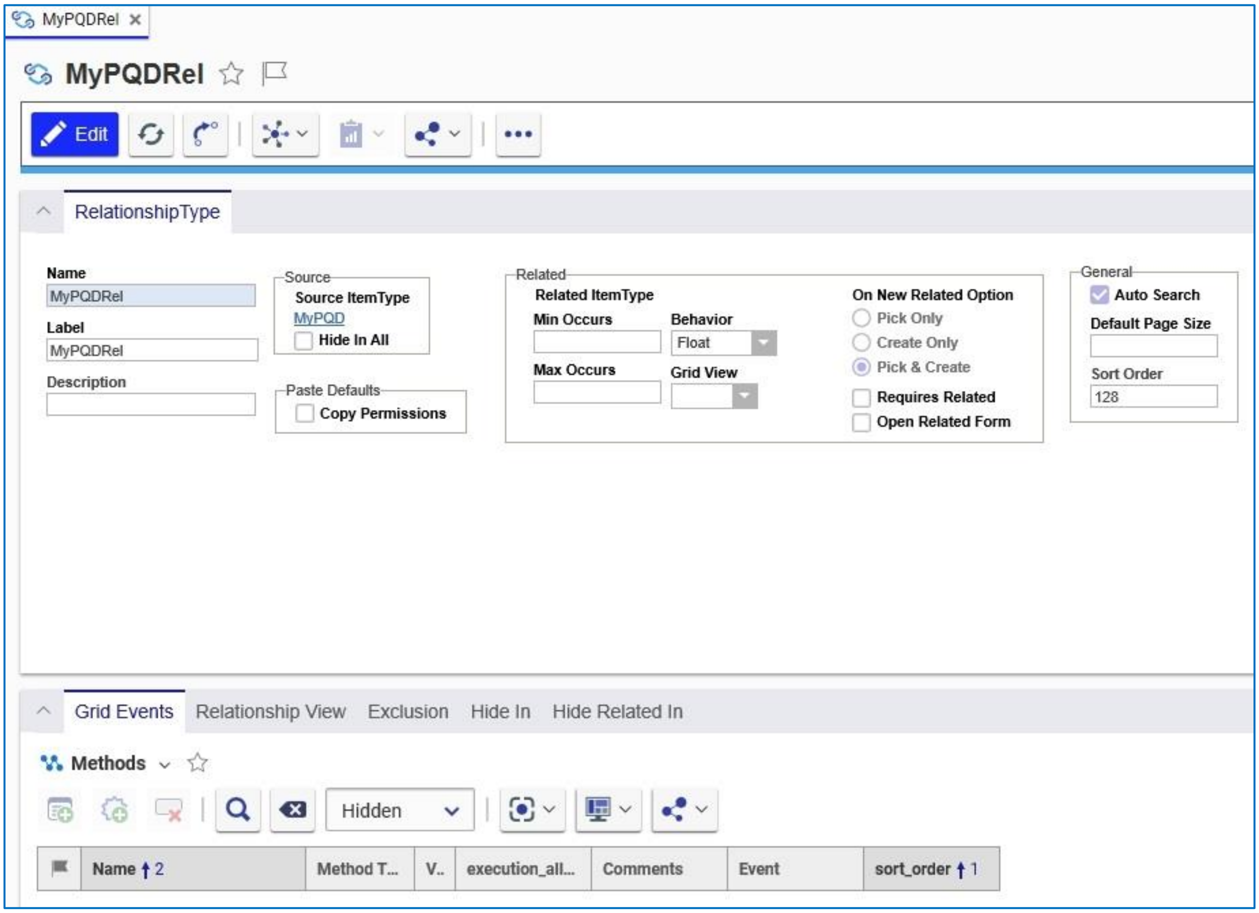
☑ Auto Search

Figure 8.

3. Enter a name and label.
    a. Name: MyPQDRel
    b. Label: MyPQDRel
4. Set the Source ItemType to the CMF ItemType: MyPQD.

5. Click [Save] and [✓ Done] to save and unclaim the item.

   It should look like Figure 9:



Name: MyPQDRel
Label: MyPQDRel
Description:

Source
Source ItemType: MyPQD
□ Hide In All

Paste Defaults
□ Copy Permissions

Related
Related ItemType
Min Occurs:
Max Occurs:
Behavior: Float
Grid View:
On New Related Option
○ Pick Only
○ Create Only
● Pick & Create
□ Requires Related
□ Open Related Form

General
☑ Auto Search
Default Page Size:
Sort Order: 128

Figure 9.

2.4 Content Type

The Content Type item links the Business Object ItemType to the CMF ItemType and is used to configure the custom grid and tree on the CMF items.

1. Navigate to **Administration → Content Modeling → Content Types** in the TOC. The following menu appears:

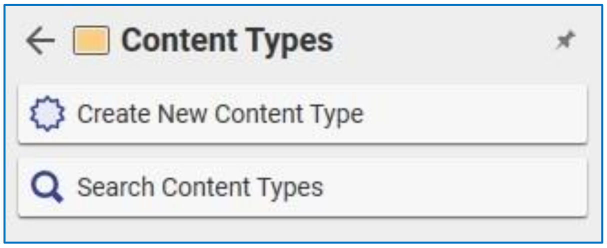


Figure 10.

2. Click **Create New Content Type**. The following screen appears:

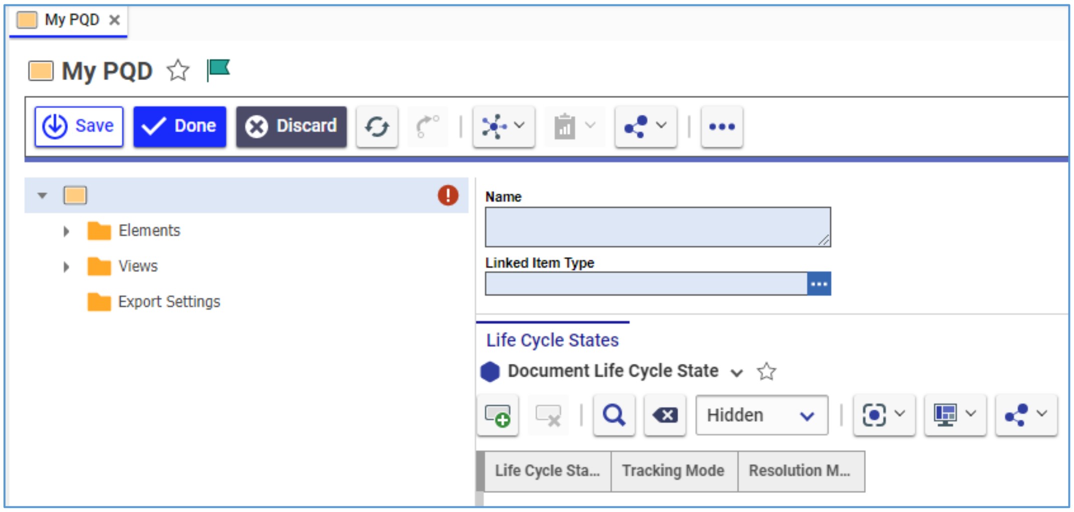


Figure 11.

3. Enter **My PQD** in the Name field.
4. Set the Linked ItemType to the CMF ItemType. In this instance, that is MyPQD.

   It should now look like Figure 12.

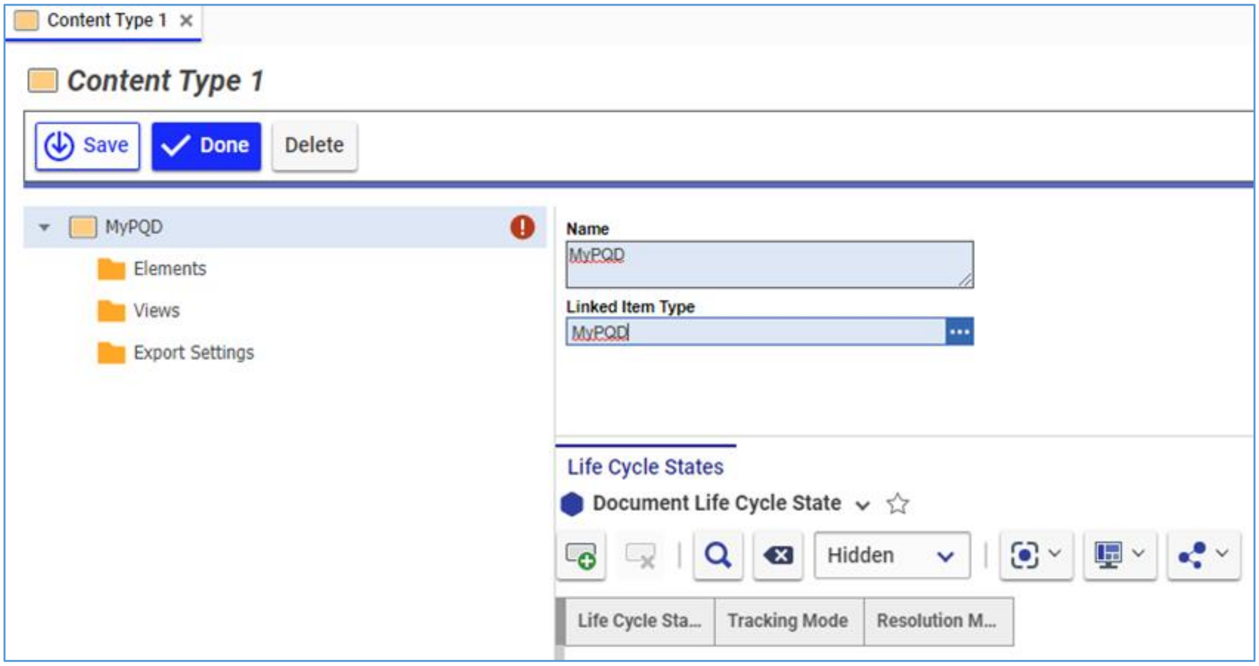
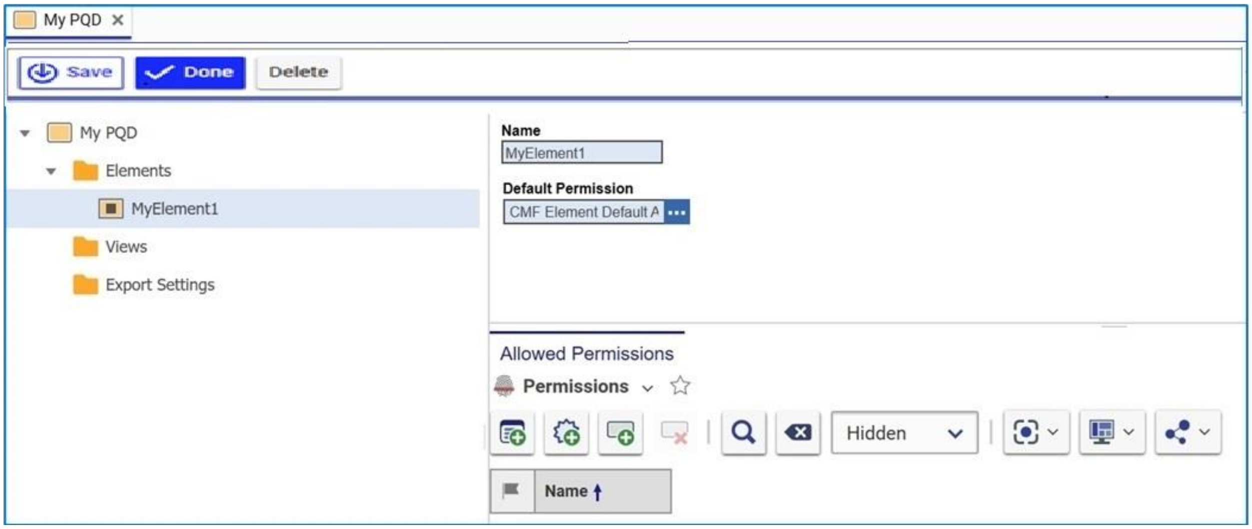


Figure 12.

5. Right-click the **Elements** tab on the Content Type tree and select **Add Element Type**.

6. Enter **MyElement1** in the Name field for the Element Type.

   It should now look like Figure 13.



Name: MyElement1

Default Permission: CMF Element Default A

Figure 13.

2.4.1 Content Type Elements

1. Right-click the Element (MyElement1) in the tree and select **Add Property Type**
2. For this example, create the following Property:
   a. Data Type: Text
   b. Name: MyTextProp
3. Click ⬇ to save the property.
4. Create another Property:
   a. Data Type: String
   b. Name: MyStringProp
   c. Data Length: 32

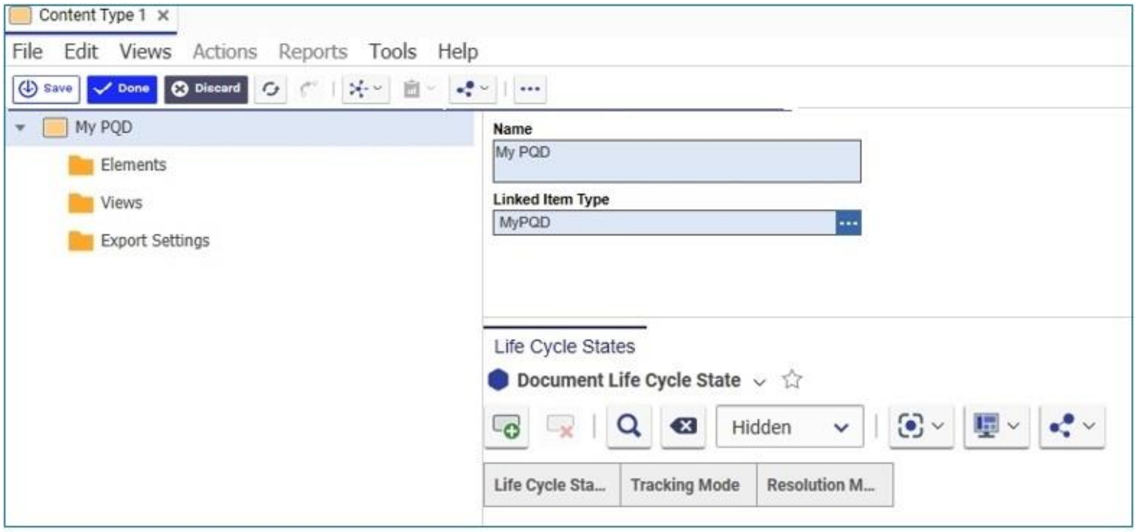
   Refer to Table 2 for a complete list of the properties of the Element Properties.

Table 1: Document Conventions

Property	Description
DataType	DataType of the Element’s property
Name	Element’s name, a string, 30
My Data Length	The maximum character length, only used for strings
Default Permission	A default **Permission** Item set for new Elements of this type, optional.

5. Save Content Type.

   It should look like Figure 14.



Name: My PQD

Linked Item Type: MyPQD

Figure 14.

Note: If there is a need to create several elements without parent (parallel structures of elements), each of those parallel structures should be organized into different views to be displayed in CMF editor. Otherwise, the CMF editor will show nothing.

6. Right-click the MyElement1 item again and select **Add Element Binding**.
7. For this example, set the following properties:
   a. Referenced Type: MyProcessPlan (Business Object ItemType)
   b. Tracking Mode: Show Differences
   c. Resolution Mode: Current
   d. New Element Mode: Pick or Create

   Refer to Table 3 for a complete list of the Element Binding items.

Table 2: Document Conventions

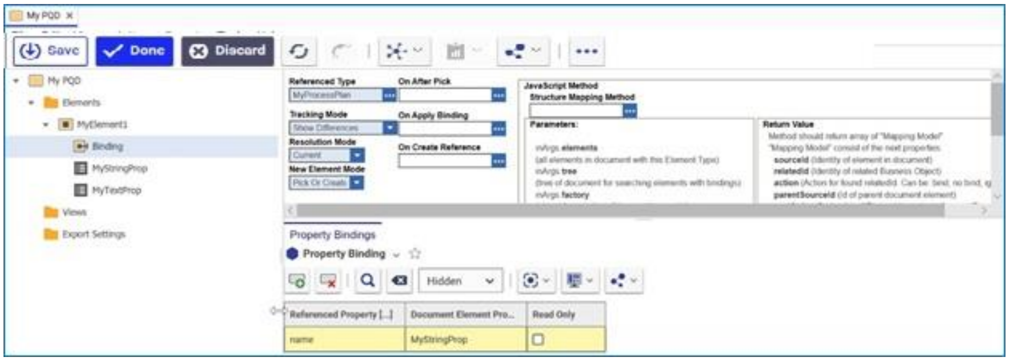
Property	Description
Referenced Type	The Business Object ItemType that will be bound to the Element Type
Tracking Mode	- Show Differences: Show differences between Elements of this type and the referenced Business Object - Ignore Differences: Ignore differences between Elements of this type and the referenced Business Object - Hard Show Differences: Overrides Content Type Life Cycle-specific tracking mode - Hard Ignore Differences: Overrides Content Type Life Cycle-specific tracking mode

Property	Description
Resolution Mode	Identifies which generation of bound item to access to perform value comparison. - Current: use the current version of the item - As Saved: use bound_item_id as is, do not use the current version of the item - Hard Current: Overrides the LifeCycle resolution mode - Hard As Saved: Overrides the LifeCycle resolution mode
New Element Mode	Define how new elements may be added. - Pick or Create - Pick Only - Create Only
Reference Is Required	User cannot create an unbounded Element. All Elements of this type must have a reference to a Business Object.
Structure Mapping Method (SMM)	The SMM returns the table data, allowing users to see conflicts between structures. The SMM is executed based on the Element Type. If a structure contains two different Element Types with SMM, both methods will run. This method is called when: - An Element is added (New, Pick, Create) - An Element is deleted - A Reference is released - On Pick/Replace of an Element
On Create Reference	Optional. This method is called when a user attempts to create a reference on a Business Object. If no method is set, the default implementation will execute.
On After Pick	A method called after a user picks some business object from the context menu.
On Apply Binding	Optional. The method will be called when the Element is picked. The method should link the selected Business Object to hierarchy of the main Business Object. This method also will be called when an Element exists but is not in the hierarchy of Business Object and the user wants to include the Element into the hierarchy of the Business Object.

8. Add a new item to the **Property Bindings** tab.
9. Configure the following properties:
    a. Referenced Property: name
    b. Document Element: MyStringProp

10. Save the Content Type.

    It should look like Figure 15.



Referenced Type: MyProcessPlan
Tracking Mode: Show Differences
Resolution Mode: Current
New Element Mode: Pick Or Create
On After Pick:
On Apply Binding:
On Create Reference:

JavaScript Method
Structure Mapping Method:

Parameters:
inArgs.**elements**
(all elements in document with this Element Type)
inArgs.**tree**
(tree of document for searching elements with bindings)
inArgs.**factory**

Return Value
Method should return array of "Mapping Model"
"Mapping Model" consist of the next properties:
sourceId (Identity of element in document)
relatedId (Identity of related Business Object)
action (Action for found relatedId. Can be: bind, no bind, e
parentSourceId (Id of parent document element)

Referenced Property [...]	Document Element Pro...	Read Only
name	MyStringProp	☐

Figure 15.

Currently, the Binding Types intersection is supported for the following Data Types:

- String
- Text
- Integer
- Date
- Boolean
- Image

2.4.2 Content Type Element Identity Inheritance

Content Type Elements inherit the managed_by_id, owned_by_id, and team_id properties from the parent Content Type instance item. The Element’s created_by_id and modified_by_id properties are not inherited.

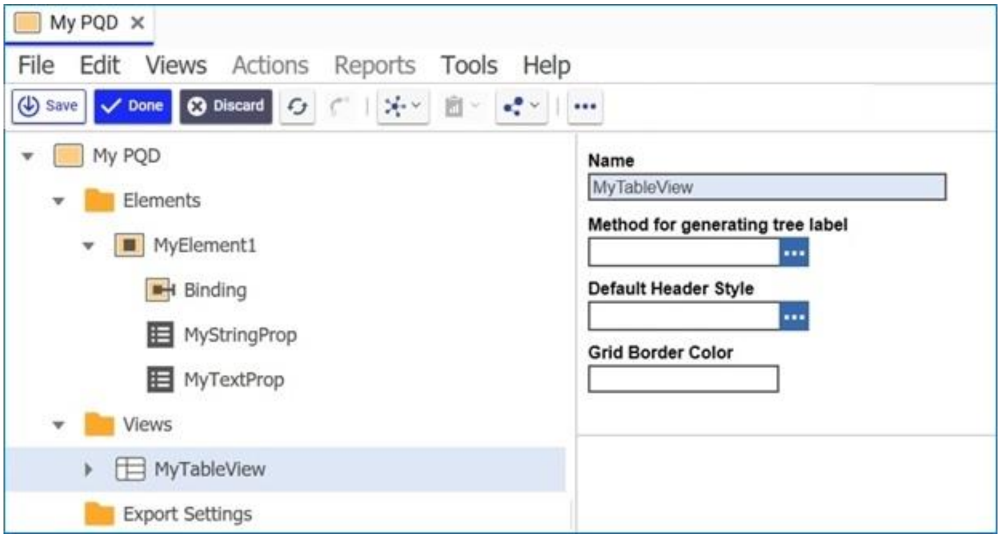
For this sample case, this means that any MyElement1 items will be created with the managed_by_id, owned_by_id, and team_id values from the parent MyPQD item.

Warning It is important to consider identity inheritance when defining custom permissions on Content Type Elements. Permissions including the Creator system identity on Content Type Elements will use the creator of each Element item, not the creator of the parent Content Type instance item, when validating user permissions.

2.4.3 Content Type View

1. Right-click the **Views** node in the Content Type tree and select **Add View**.
2. Click **OK** to confirm a tabular view.
3. Enter a name for the Tabular View.

   It should look like Figure 9.



Name: MyTableView

Method for generating tree label

Default Header Style

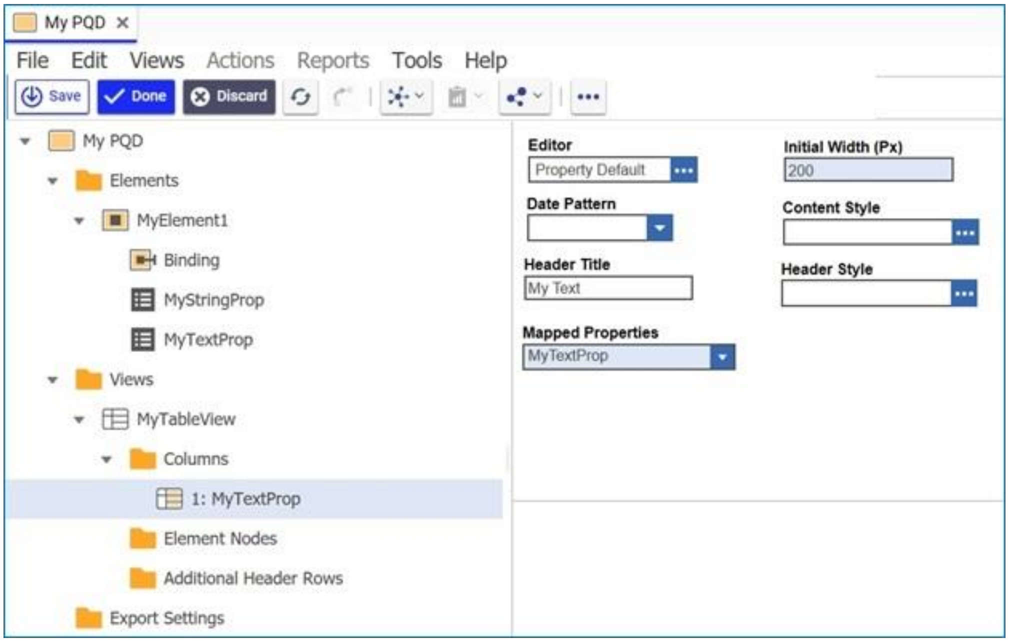
Grid Border Color

Figure 16.

4. Expand the Tabular View sub-tree
5. Right-click the **Columns** node and select **Add Column**.
6. Enter the following values in the new Tabular View Column
   a. Mapped Property: MyStringProp
   b. Header Title: My String
   c. Initial Width: 100

7. Add another Column to the Columns node.
    a. Mapped Property: MyTextProp
    b. Header Title: My Text
    c. Initial Width: 200

    It should look like Figure 17.



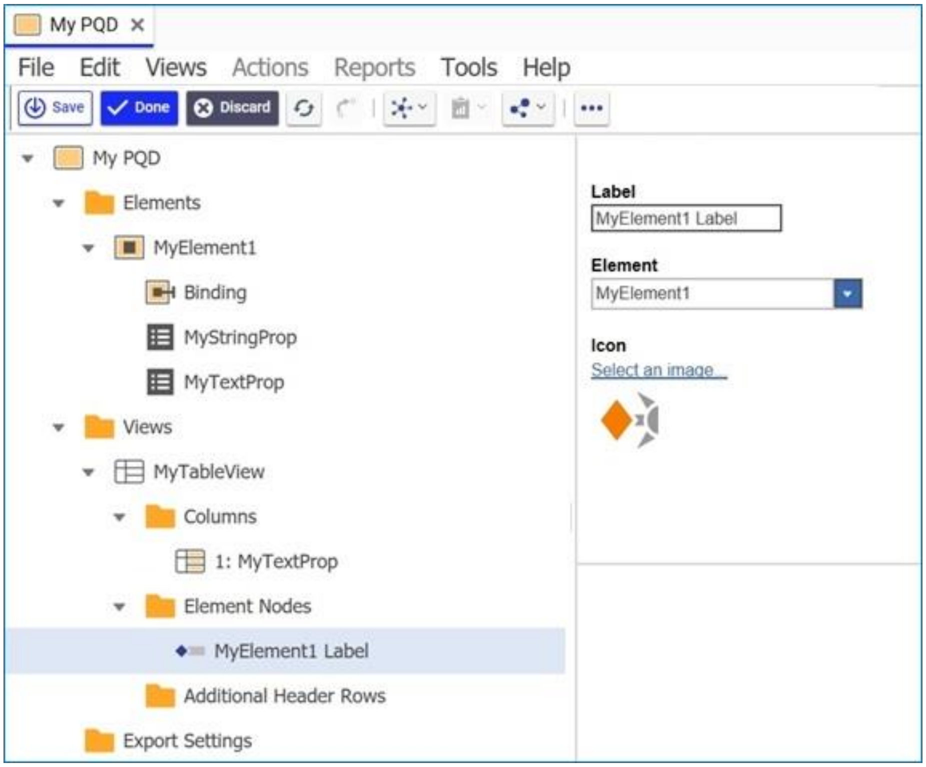
Editor: Property Default
Initial Width (Px): 200
Date Pattern:
Content Style:
Header Title: My Text
Header Style:
Mapped Properties: MyTextProp

Figure 17.

8. Right-click the **Element Nodes** node on the Tabular View sub-tree and select **Add Element Type Configuration**.

9. Enter the following properties for the Element Node:
    a. Label: MyElement1 Label
    b. Element Type: MyElement1
    c. Icon: Choose an image

    It should look like Figure 18.



Label: MyElement1 Label

Element: MyElement1

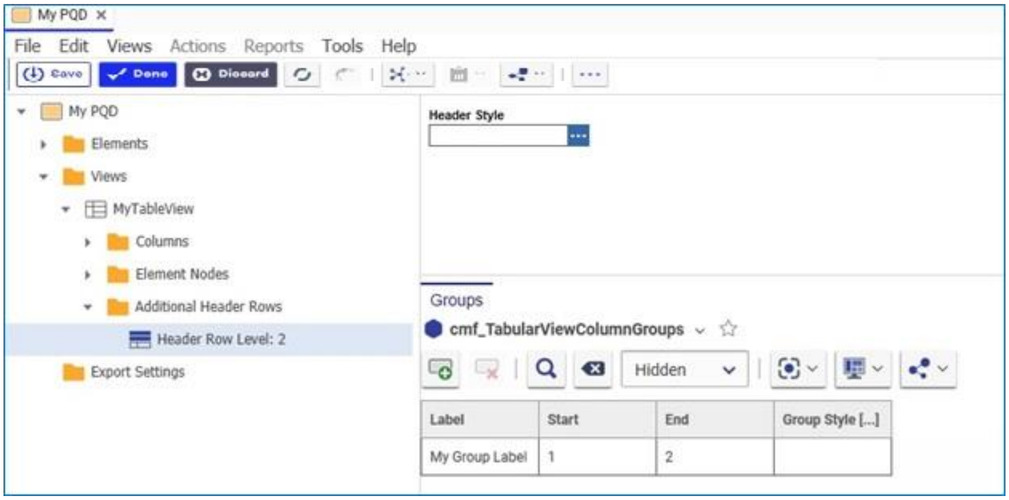
Icon

Figure 18.

10. Right-click the Additional Header Rows node and select **Add Additional Header Row**.

11. Add a new Group relationship with the following properties:
    a. Label: My Group Label
    b. Start: 1
    c. End: 2

    It should look like Figure 19.



Label	Start	End	Group Style [...]
My Group Label	1	2	

Figure 19.

12. Save, Unclaim, & Close the Content Type.

2.5 Viewing the Linked ItemType

Use the following procedure to confirm the configuration of the Business Object ItemType and the linked Content Type.

1. Create an instance of the Business Object.
    a. Click **My Process Plans** in the TOC. The following menu appears:

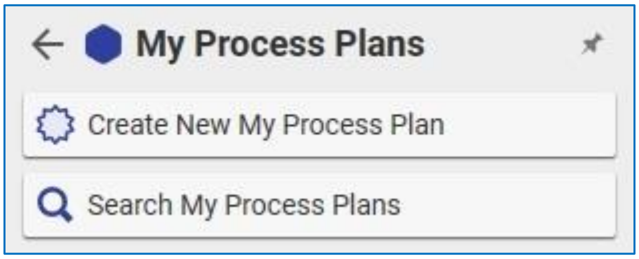
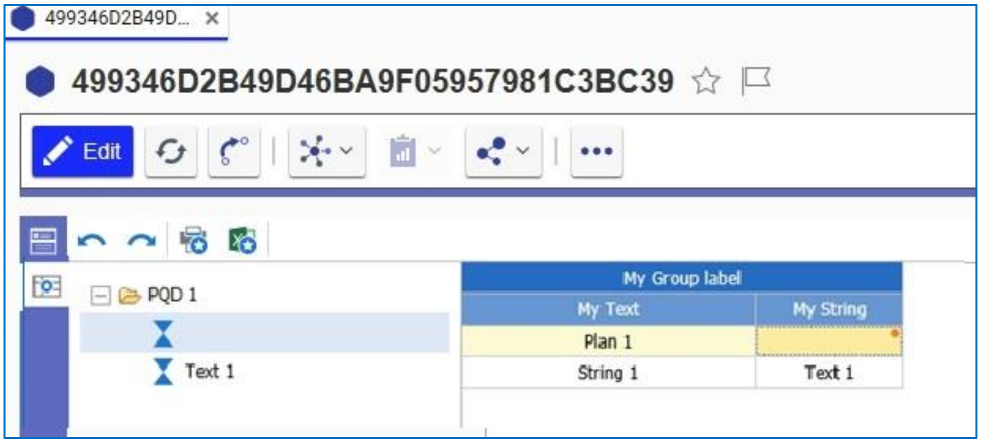


Figure 20.

    b. Click **Create New MyProcessPlan** and create a new My Process Plan item with the name “Plan 1”.
    c. Click **Done**.

2. Create an instance of the CMF Object.
   a. Navigate to My PQDs in the TOC.
   b. Create a new My PQD item.
   c. Set name as “PQD 1”.
   d. Save PQD 1.
3. In the CMF Item, click the **Show Editor** icon in the sidebar.
4. Add an unbounded element to the Content ItemType:
   a. Right-click the root element tab in the tree and select **Insert MyElement1 Label → New.**
   b. Enter values for **My String** and **My Text**.
5. Add a bound element from reference:
   a. Right-click the root element tab in the tree and select **Insert MyElement1 Label → From Reference → Select**.
   b. Run a search in the Search Dialog, select “Plan 1”, and return the selected object (click the green checkmark).

   The tree and grid should appear configured as in Figure 21:



My Group label	
My Text	My String
Plan 1	
String 1	Text 1

Figure 21.

2.6 Exporting CMF Packages

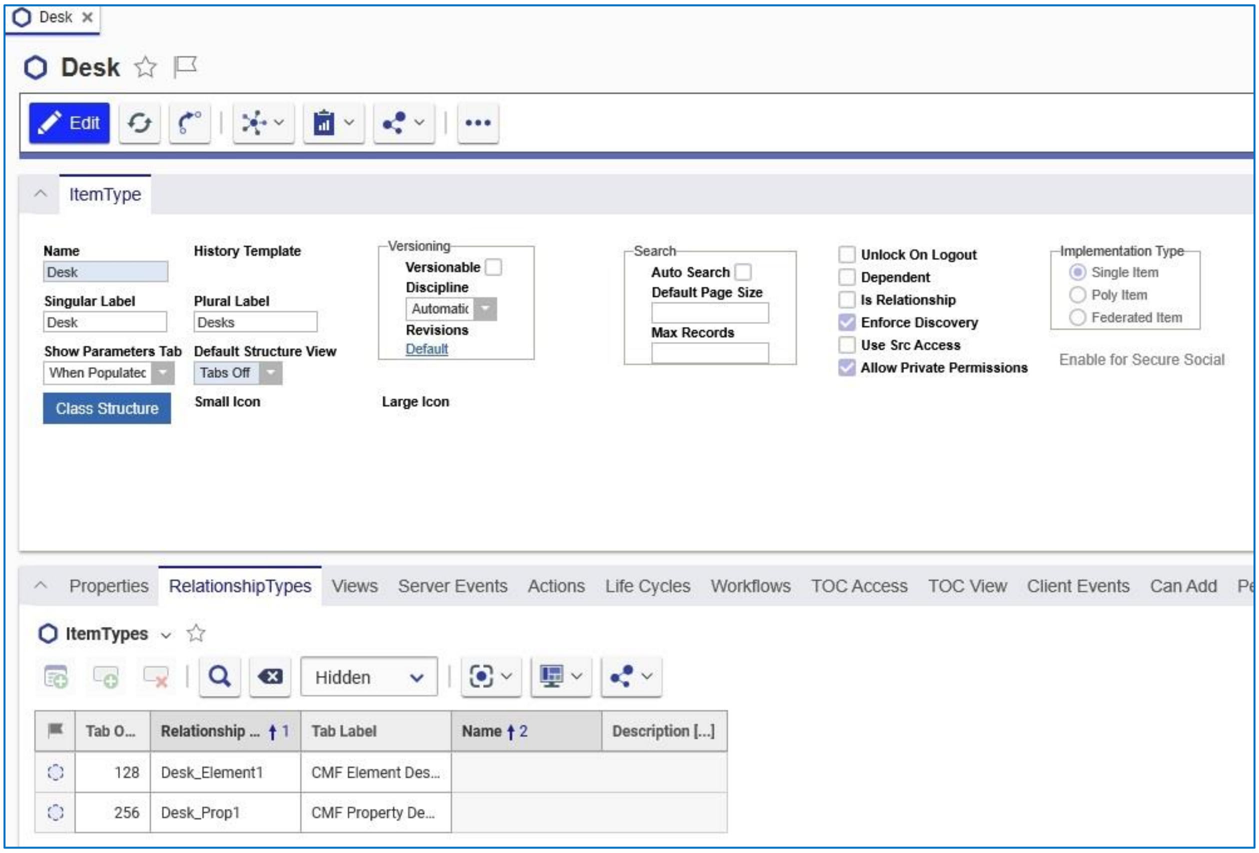
Warning When creating and exporting packages that include CMF Content Types, it is important to keep in mind that there may be RelationshipTypes that are automatically created during the CMF process. These RelationshipTypes should not be added or exported. Doing so will cause import errors.

To see an example of these RelationshipTypes, create the following:

1. Create a new ItemType called “Desk.”
2. Create a new Content Type item:
   a. Name = “Desk_CMF”.
   b. Linked Item Type = “Desk”.

3. Add an Element to Desk_CMF called Desk_Element1.
4. Add a Property to the Desk_Element1 called Desk_Prop1.
5. Save and close the Desk_CMF Content Type.
6. Open the Desk ItemType and go to the RelationshipType tab.

There should be 2 items that were automatically created in the CMF process.



Name: Desk
History Template:
Singular Label: Desk
Plural Label: Desks
Show Parameters Tab: When Populatec
Default Structure View: Tabs Off

Versioning
Versionable □
Discipline: Automatic
Revisions: Default

Search
Auto Search □
Default Page Size:
Max Records:

□ Unlock On Logout
□ Dependent
□ Is Relationship
☑ Enforce Discovery
□ Use Src Access
☑ Allow Private Permissions

Implementation Type
● Single Item
○ Poly Item
○ Federated Item

	Tab O...	Relationship ... ↑ 1	Tab Label	Name ↑ 2	Description [...]
	128	Desk_Element1	CMF Element Des...		
	256	Desk_Prop1	CMF Property De...		

Figure 22.

Note: These RelationshipType items ***should not*** be added to Package Definitions for exporting. When exporting the CMF, it is necessary to include the cmf_BaseView item for the CMF to load properly. Navigate to the cmf_BaseView ItemType and expose it in the TOC by giving it the correct TOC Access settings. Find the cmf_BaseView item associated with your CMF ItemType and add it to the package definition.

When importing the Content Type item, these RelationshipType items will be created automatically by default.

3 Printing and Exporting CMF Items

The Content Modeling Framework includes options to Print content and/or export the content to an Excel spreadsheet. This part of the solution is available to Aras subscribers.

3.1 Required Installations

The following needs to be installed to publish Technical Documents:

- Aras Conversion Server; see *Aras Innovator 2023 Release - Conversion Server Setup Guide*.
- Aras Agent Service; see Section 5.12 of the *Aras Innovator 2023 Release – Installation Guide*.

3.2 Feature License Key

The Aras.CMF license comes as part of most license bundles for subscribers, such as Aras.PremierSubscription. It does not require a separate activation. To check whether a valid **Feature License** exists in Aras Innovator:

1. Log into Aras Innovator as an administrator.
2. Go to **Administration** --> **Feature Licenses**.
3. If a subscription license is found, check the expiration date to confirm that the license is still active.

3.2.1 Requesting a Feature License

In order to request a feature license activation key, customer with an active Aras subscription can send an email to licenses@aras.com or to your account representative in the following format:

- Subject: Aras CMF Activation Key Required
- Body: Version of Aras Innovator

You will receive a reply containing the Feature License activation key.

3.2.2 Installing the License

Once you have the activation key, install the feature license as follows:

1. Log into Aras Innovator as an administrator.

2. Click the user Menu icon IA. A dropdown menu appears:

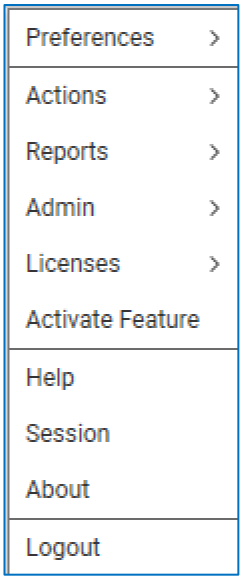


Figure 23.

3. Click Activate Feature:

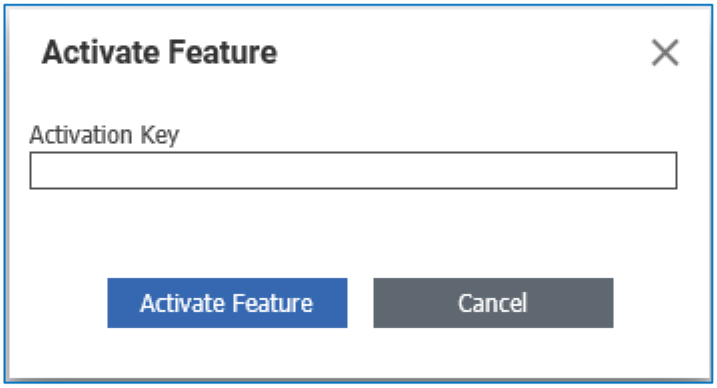


Figure 24.

4. Paste the acquired activation key and click **Activate Feature**.

   A success message should pop up.

3.3 Enabling Printing and Exporting Options

The Aras Publishing Service feature requires that the Conversion Server be configured to enable it.

To enable the service:

1. Open the ConversionServerConfig.xml file for editing.

   Typically, this file is located at the root of Aras Innovator code tree (Sample: `C:\Program Files (x86)\Aras\Innovator\ConversionServerConfig.xml`)

2. Add the following into the section `<Converters/>`:

   ```
   <Converter name="cmf_ExcelPublishingConverter"
   type="Aras.Cmf.Publishing.Excel.ExcelExportConverter,
   Aras.Cmf.Publishing" />
   ```

The following is a sample of a ConversionServerConfig.xml file with the required setup:

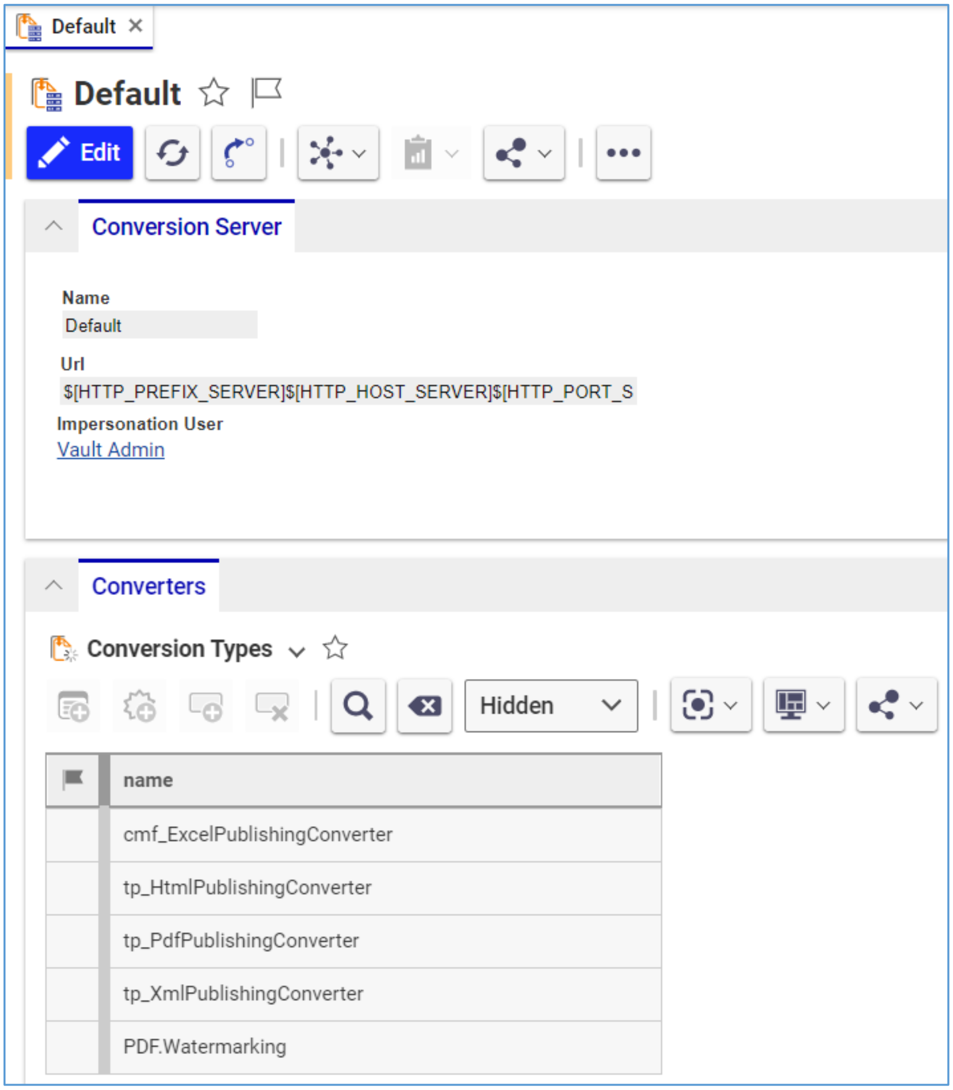
```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <configSections>
       <!-- Common converter service configuration -->
       <section name="ConversionServer"
type="Aras.ConversionFramework.ConversionServer.Configuration.ConversionServe
rConfigurationSection, Conversion.Base"/>
    <sectionGroup name="ConverterSettings">
    <!-- Place here class configuration section definitions for converters --
>
    </sectionGroup>
    </configSections>
    <ConversionServer>
   <InnovatorServer
url="http://vabraham02/support/Solutions120SP3/Server/InnovatorServer.aspx"/>
       <Converters>
       <Converter name="tp_XmlPublishingConverter"
type="Aras.Publishing.XmlPublishingConverter, Aras.TDF.PublishingConverter"/>
       <Converter name="tp_HtmlPublishingConverter"
type="Aras.Publishing.HtmlPublishingConverter,
Aras.TDF.PublishingConverter"/>
     <Converter name="cmf_ExcelPublishingConverter"
type="Aras.Cmf.Publishing.Excel.ExcelExportConverter, Aras.Cmf.Publishing" />
                        </Converters>
      </ConversionServer>
      <ConverterSettings>
  <!-- Place here configuration sections for converters -->
       </ConverterSettings>
</configuration>
```

3. Save and close the `ConversionServerConfig.xml` file.

3.4 Managing Conversion Rules

By default, the conversion rules required for CMF are added to the **Default** Conversion Server. If your Aras Innovator setup requires the use of a different Conversion Server, the following Conversion Rules should be added to the **Converters** tab on the appropriate **Conversion Server** item in the database:

- cmf_ExcelPublishingConverter



Name
Default

Url
$[HTTP_PREFIX_SERVER]$[HTTP_HOST_SERVER]$[HTTP_PORT_S

Impersonation User
Vault Admin

	name
	cmf_ExcelPublishingConverter
	tp_HtmlPublishingConverter
	tp_PdfPublishingConverter
	tp_XmlPublishingConverter
	PDF.Watermarking

Figure 25.

4 Content Modeling Framework (CMF) API

This section describes how to use the CMF CRUD API to carry out common tasks related to the programmatic handling of CMF documents. The tasks and solutions are given for an imaginary example of a simple CMF document that is described in the following section. It is used in all the examples that follow.

4.1 Sample CMF Document Parts Warehouse

Our sample document describes an imaginary warehouse that is partitioned into Zones, each of which is subdivided into Places. Each Place can contain one or more Parts items. The following data model diagram illustrates the structure of the document.

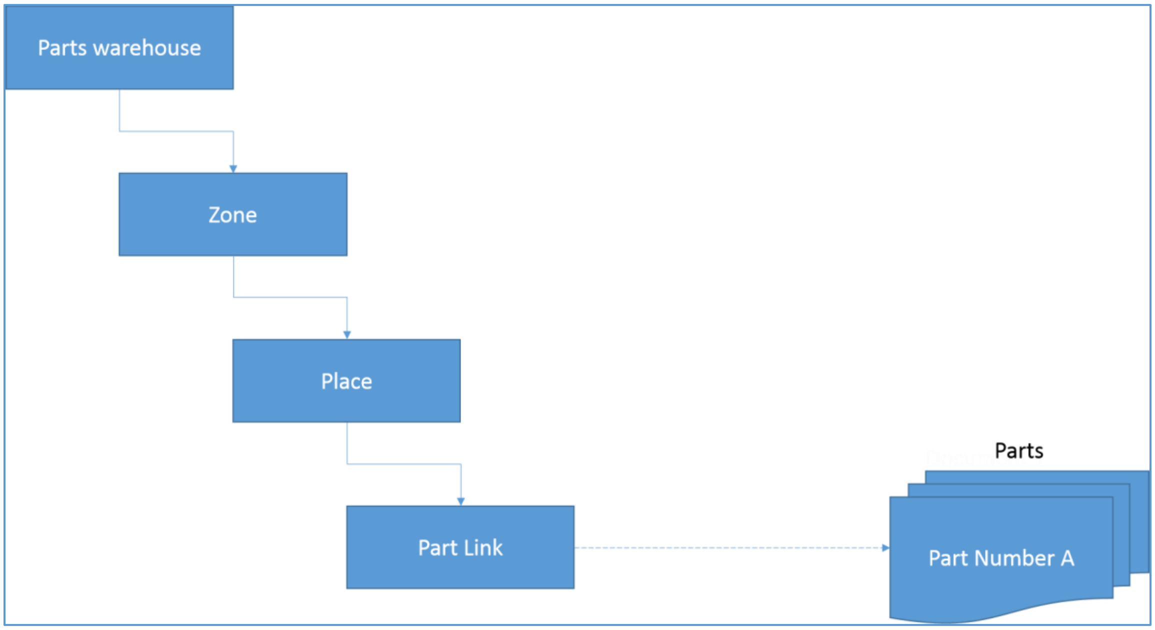
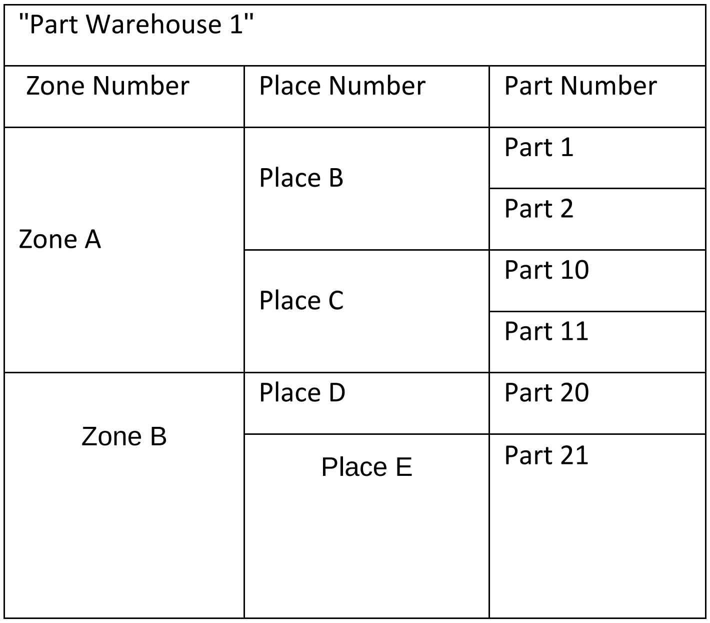


Figure 26.

The code examples used in the following sections assume that a Parts Warehouse CMF Document has been appropriately configured in Aras Innovator.

4.1.1 Sample Warehouse

Figure 27 shows an example of the tabular content as it would be rendered in the CMF Document Viewer. It includes content describing a small warehouse that we will be using in our code samples.



"Part Warehouse 1"		
Zone Number	Place Number	Part Number
Zone A	Place B	Part 1
		Part 2
	Place C	Part 10
		Part 11
Zone B	Place D	Part 20
	Place E	Part 21

Figure 27.

Zones, Places and Parts are identified by their “numbers” – a string property that contains a “name” / “identifier” of an item. For instance, in the sample warehouse above the warehouse named "Part Warehouse 1" contains two zones “Zone A” and “Zone B.”

4.1.2 Loading a document using the cmf_addElement action

Task:

Load the CMF Document data from an external source (jsonObject).

Solution description:

We are going to write a method that, given a Parts Warehouse CMF Document and the document content in JSON format (method parameter names documentId and jsonDocument respectively), loads the document content into the Parts Warehouse CMF Document. The Parts Warehouse CMF Document is assumed to be empty at the start of the method execution. We will use part numbers to look for related parts in Aras Innovator in the json document.

Here is what our sample warehouse data looks like in JSON format:

```
{
   "zones" : [
   {
      "number" : "Zone A",
      "places" : [
       {
         "number": "Place B",
         "partLinks" : [
          {
             "partNumber" : "Part 1"
          },
          {
             "partNumber" : "Part 2"
          }
          ],
       },
       {
         "number": "Place C",
         "partLinks" : [
          {
            "partNumber" : "Part 10"
     },
     {
       "partNumber" : "Part 20"
     }
         ],
      }]
   }]
}
```

Method:

JavaScript

Note: You need to have “can_get” and “can_update” permissions on the document to run the following code.

```
function loadCMFDocument(documentId, jsonDocument) {
   var innovator = aras.newIOMInnovator();
   for (var i = 0; i < jsonDocument.zones.length; i++) {
      var zone = jsonDocument.zones[i];

      // we need to have identifier to connect elements between each other
(parent-child)
      var zoneId = aras.generateNewGUID();

      // create "zone" cmf element using "cmf_addElement" action
      createZoneElement(zone, zoneId);
      var places = zone.places;
      for (var j = 0; j < places.length; j++) {
         var place = places[j];
         var placeId = aras.generateNewGUID();

         // create "place" cmf element using "cmf_addElement" action
         createPlaceElement(place, placeId, zoneId);

         for (var k = 0; k < place.partLinks.length; k++) {
            var partLink = place.partLinks[k];
            // create "part link" cmf element using "cmf_addElement" action
            createPartLinkElement(partLink, placeId);
         }
      }
   }

   function createZoneElement(zone, id) {
      var element = innovator.newItem("Zone", "cmf_addElement");
      element.setAttribute("id", id);
      element.setProperty("Zone_Number", zone.number);
      element.setProperty("document_id", documentId);
      element.apply();
   }

   function createPlaceElement(place, id, parentId) {
      var element = innovator.newItem("Place", "cmf_addElement");
      element.setAttribute("id", id);
      element.setProperty("Place_Number", place.number);
      element.setProperty("document_id", documentId);
      element.setProperty("parent_element_id", parentId);
      element.apply();
   }

   function createPartLinkElement(partLink, parentId) {
      // find part identifier by part number into innovator
      var partId = getPartId(partLink);
      if (partId) { // if didn’t find it then ignore
         var element = innovator.newItem("Part Link", "cmf_addElement");
         element.setProperty("bound_item_id", partLink.id);
         element.setProperty("document_id", documentId);
         element.setProperty("parent_element_id", parentId);
         element.apply();
      }
   }

           function getPartId(partLink) {
              var partRequest = innovator.newItem("Part", "get");
              partRequest.setProperty("item_number", partLink.partNumber);
              var partResponse = partRequest.apply();
              if (partResponse.isError()) {
                 // we didn't find it or we haven't permissions
                 return null;
              } else {
                 return partResponse.getID();
              }
   }
}
```

4.1.3 Reading a document using the cmf_getDocument action

Task:

Export the existing CMF document into an external datasource (jsonObject).

Solution description:

Write a method that returns a JSON object containing the content for a given Parts Warehouse CMF Document (method paramenter name *documentId*).

Note: You need to have “can_get” permissions on the document to to run the following code.

In the following example, the returned JSON object looks like this:

```
{
   "zones" : [
   {
      "number" : "Zone A",
      "places" : [
       {
         "number": "Place B",
         "partLinks" : [
          {
             "partId" : "{ID OF LINKED PART 1}",
             "partNumber": "Part 1"

          },
          {
             "partId" : "{ID OF LINKED PART 2}",
              "partNumber": "Part 2"
          }
          ],
       },
       {
         "number": "Place C",
         "partLinks" : [
          {
            "partId" : "{ID OF LINKED PART 10}",
            "partNumber": "Part 10"
     },
     {
       "id" : "{ID OF LINKED PART 11}"
             "partNumber": "Part 11"
     }
          ],
       }]
   }]
```

Method:

JavaScript

```
function documentToJSON(documentId) {
   var innovator = aras.newIOMInnovator();
   // get all cmf document ("Parts warehouse") using "cmf_getDocument" action
   var documentItem = innovator.newItem("Parts warehouse",
"cmf_getDocument");
   documentItem.setAttribute("id", documentId);
   var response = documentItem.apply();

   if (response.isError()) {
      aras.AlertError(response.getErrorDetail());
      return;
   } else {
      // initiate result json object
      var resultDocument = {};
      resultDocument.zones = [];

     var zoneRelationships = response.getRelationships("Zone");
     for (var i = 0; i < zoneRelationships.getItemCount() ; i++) {
        // fill zones, their places and part links
        var zoneObject =
createZoneObject(zoneRelationships.getItemByIndex(i));
        resultDocument.zones.push(zoneObject);
     }
     // convert json object into string
     return JSON.stringify(resultDocument);
   }

   function createZoneObject(zone) {
      var zoneNumber = zone.getProperty("Zone_Number");
      var zoneId = zone.getAttribute("id");

      var zoneObject = { id: zoneId, zoneNumber: zoneNumber, places: [] };
      var placeRelationships = zone.getRelationships("Place");
      // fill "places" for particular "zone"
      for (var j = 0; j < placeRelationships.getItemCount() ; j++) {
         var placeObject =
createPlaceObject(placeRelationships.getItemByIndex(j));
         zoneObject.places.push(placeObject);
      }
      return zoneObject;
   }

   function createPlaceObject(place) {
      var placeNumber = place.getProperty("Place_Number");
      var placeId = place.getAttribute("id");
      var placeObject = { id: placeId, placeNumber: placeNumber, parts: [] };
      var partLinkRelationships = place.getRelationships("Part Link");
      // fill "part links" for particular "place"
      for (var k = 0; k < partLinkRelationships.getItemCount() ; k++) {
         var partObject =
createPartLinkObject(partLinkRelationships.getItemByIndex(k));
         placeObject.parts.push(partObject);
      }
      return placeObject;
   }

   function createPartLinkObject(partLink) {
      var partNumber = partLink.getProperty("Part_Number");
      var partId = partLink.getProperty("bound_item_id");
      return { partId: partId, partNumber: partNumber };
   }
}
```

4.1.4 Reading a Document Element Using the cmf_getElement action

4.1.4.1 Counting Elements in a CMF Document

Task:

Find how many Places are in a specific Zone.

Solution description:

We are going to write a method that will return the total number of Place Document Elements that are included in a given Zone (method paramenter name `zoneId`).

To find, how many Places are in a Zone we will create a request that returns all zone places and then we will count them.

Note: You need to have “can_get” permissions on the document to to run the following code.

Method:

JavaScript

```
function getPlaceCount(zoneId) {
   var innovator = aras.newIOMInnovator();

   var zoneElement = innovator.newItem("Zone", "cmf_getElement");
   zoneElement.setAttribute("id", zoneId);

   // add child "place" element as a relationship
   var placeElement = innovator.newItem("Place");
   zoneElement.addRelationship(placeElement);

   var response = zoneElement.apply();
   if (response.isError()) {
      aras.AlertError(response.getErrorDetail());
      return;
   } else {
      // get all places for the found "zone"
      var relationships = response.getRelationships();
      return relationships.getItemCount();
   }
}
```

4.1.5 Reading nested elements in the document

Task:

Obtain all "Part Links" for a given zone.

Solution description:

To get all "Part Links" for a given zone, you need to fetch all the zone "Places" and their "Part Links.” To do this you need to create a request using the following structure:

```
<Item type="Zone" id="..." action="cmf_getElement">
   <Relationships>
      <Item type="Place">
         <Relationships>
            <Item type="Part Link" />
         </Relationships>
      </Item>
   </Relationships>
</Item>
```

You are going to write a method that returns the Part Links for a given zone (method parameter name zoneId) returns its Part Links.

Note: You need to have “can_get” permissions on the document to to run the following code.

Method:

JavaScript

```
function getPartLinks(zoneId) {
   var innovator = aras.newIOMInnovator();
   // get all places and part links for particular zone
   var zoneElement = innovator.newItem("Zone", "cmf_getElement");
   zoneElement.setAttribute("id", zoneId);
   var placeElement = innovator.newItem("Place");
   var partLinkElement = innovator.newItem("Part Link");
   placeElement.addRelationship(partLinkElement);
   zoneElement.addRelationship(placeElement);
   var response = zoneElement.apply();

   if (!response.isError()) {
      // we can use response.getItemsByXPath("//Item[@type='Part Link']");
      // or use loops as shown below
      var partLinks = [];
      var placeRelationships = response.getRelationships();
      for (var i = 0 ; i < placeRelationships.getItemCount() ; i++) {
         var place = placeRelationships.getItemByIndex(i);
         var partLinkRelationships = place.getRelationships();
         for (var j = 0; j < partLinkRelationships.getItemCount() ; j++) {
            partLinks.push(partLinkRelationships.getItemByIndex(j));
         }
      }
      if (partLinks.length > 0) {
         var resultItem = partLinks[0];
         for (var k = 1; k < partLinks.length; k++) {
            resultItem.appendItem(partLinks[k]);
         }
         return resultItem;
      }
      else {
       return [];
      }
    } else {
       aras.AlertError(response.getErrorDetail());
       return [];
    }
}
```

4.1.6 Reading elements in the document using condition attributes

Task:

Obtain Places that have Part Links where number starts from “Part 1.”

Solution description:

Use the same request from the previous task but add a condition on the “Part Number” property of the “Part Link” item.

```
<Item type="Zone" id="..." action="cmf_getElement">
   <Relationships>
      <Item type="Place">
         <Relationships>
            <Item type="Part Link">
<Part_Number condition="like">Part 1%</Part_Number>
    </Item>
         </Relationships>
      </Item>
   </Relationships>
</Item>
```

Write a method that returns the Part Links for a given zone: (method paramenter name `zoneId)` returns its Part Links.

Note: You need to have “can_get” permissions on the document to to run the following code.

Method:

JavaScript

```
function getPartLinks(zoneId) {
   var innovator = aras.newIOMInnovator();
   // get all places and part links for particular zone
   var zoneElement = innovator.newItem("Zone", "cmf_getElement");
   zoneElement.setAttribute("id", zoneId);
   var placeElement = innovator.newItem("Place");
   var partLinkElement = innovator.newItem("Part Link");
   partLinkElement.setProperty("Part_Number", "Part 1%");
   partLinkElement.setPropertyAttribute("Part_Number", "condition", "like");
   placeElement.addRelationship(partLinkElement);
   zoneElement.addRelationship(placeElement);

var response = zoneElement.apply();
   if (!response.isError()) {
      // we can use response.getItemsByXPath("//Item[@type='Place']");
      var places = [];
      var placeRelationships = response.getRelationships();
      for (var i = 0 ; i < placeRelationships.getItemCount() ; i++) {
         var place = placeRelationships.getItemByIndex(i);
         places.push(place);
      }
      return places;
    } else {
       aras.AlertError(response.getErrorDetail());
       return [];
    }
}
```

4.1.7 Updating a Document Element Using the cmf_updateElement Action

4.1.7.1 Updating an element by condition

Task:

Update all zone numbers containing the specific substring: add "Restricted_" to the beginning of the "zone number".

Solution description:

First, you need to get all "zone" elements in the document. You can then loop through the zone list to see if a zone number matches the given string and, if so, change the zone number.

For example the following is the result of the document modification if the match string is “A”.

Date Before

"Part Warehouse 1"		
Zone Number	Place Number	Part Number
Zone A	…	…
		…
	…	…
		…
Zone B	…	…
	…	…

Date After

"Part Warehouse 1"		
Zone Number	Place Number	Part Number
Restricted_Zone A	…	…
		…
	…	…
		…
Zone B	…	…
	…	…

Figure 28.

Write a method that, given a document Id and a match string (method parameter names `documentId` and `template` respectively), performs the document modification.

Note: You need to have “can_get” and “can_update” permissions on the document to run the following code.

Method:

JavaScript

```
function makeZonesRestricted(documentId, template) {
   var innovator = aras.newIOMInnovator();

   // get all zones for the specific document
   var zoneElement = innovator.newItem("Zone", "cmf_getElement");
   zoneElement.setProperty("document_id", documentId);
   var response = zoneElement.apply();

   if (response.isError()) {
      aras.AlertError(response.getErrorDetail());
      return;
   } else {
      // in the loop try to find "zone numbers" which contains
substring(template)
      for (var i = 0; i < response.getItemCount() ; i++) {
         var item = response.getItemByIndex(i);
         var zoneNumber = item.getProperty("Zone_Number");
         if (zoneNumber.indexOf(template) > -1) {
            // update zone, add "Restricted_" prefix
            item.setAction("cmf_updateElement");
            item.setProperty("Zone_Number", "Restricted_" + zoneNumber);
            item.apply();
         }
      }
   }
}
```

4.1.8 Update an element binding

Task:

Change element binding from one business object to another.

Solution description:

To update an element in the document you need to know the following:

Table 1: The document identifier

Table 2: The identifier of the element that you want to update

Table 3: The new business object identifier

In this example the element is "Part Link" and the business object is "Part.”

You are going to write a method that, given a document, a part link, and a new part identifier (method parameter names documentId, partLinkId and newPartId respectively), will replace the part that the part link points to with the new part. Figure 29 shows the modifications that are performed by the method.

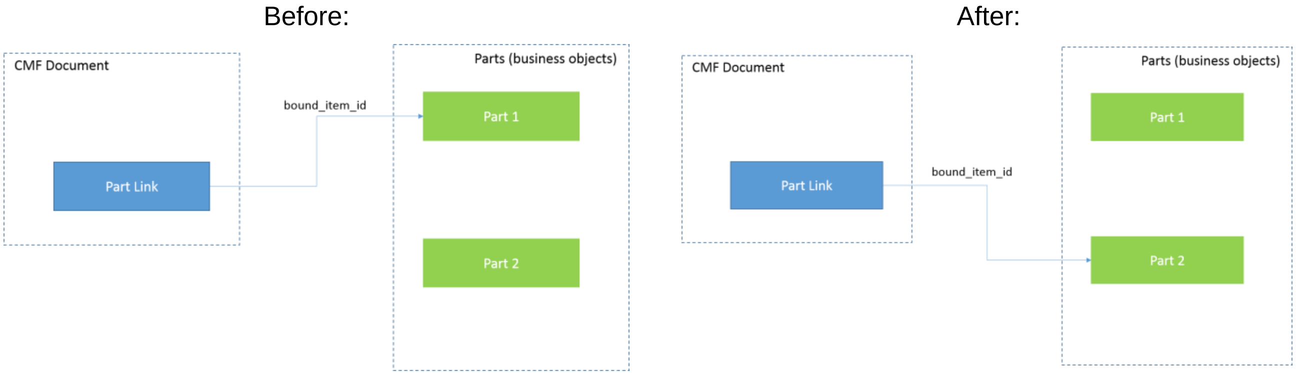


Figure 29.

Note: You need to have “can_get” and “can_update” permissions on the document to run the following code.

Method:

JavaScript

```
function replacePartLink(documentId, partLinkId, newPartId) {
   var innovator = aras.newIOMInnovator();
   var partLinkElement = innovator.newItem("Part Link", "cmf_updateElement");
   partLinkElement.setAttribute("id", partLinkId);
   partLinkElement.setProperty("document_id", documentId);
   partLinkElement.setProperty("bound_item_id", newPartId);
   partLinkElement.apply();
}
```

4.1.9 Updating an element property style

Task:

Set the text color for a given zone.

Solution description:

To set the style for an element property, you need to update the element.

Note: For the property style update to take effect, the property must be set to a new value (it can be the same as an old). Otherwise, the style will not be updated.

Write a method that, given a document, a zone, and a color (method parameter names `documentId`, `zoneId` and `color` respectively), changes the text color of a zone to a given color. (Specify the color using standard hexidecimal Web notation. For instance, "#44ff00" is a green color).

Note: You need to have “can_get” and “can_update” permissions on the document to run the following code above.

Method:

JavaScript

```
function updateStyle(documentId, zoneId, color) {
   var innovator = aras.newIOMInnovator();
   var zoneElement = innovator.newItem("Zone", "cmf_updateElement");
   zoneElement.setAttribute("id", zoneId);
   zoneElement.setProperty("document_id", documentId);

   // create cmf style object and set available styles
   var cmfStyle = innovator.newItem("cmf_Style");
   cmfStyle.setProperty("text_color", color);
   zoneElement.setPropertyAttribute("Zone_Number", "style",
cmfStyle.toString());

   // you should set value of property directly, otherwise it will be updated
to empty value
   // for this purpose we need to get existing zone element with Zone Number
property
   var existZoneElement = innovator.newItem("Zone", "cmf_getElement");
   existZoneElement.setAttribute("id", zoneId);
   existZoneElement = existZoneElement.apply();
   zoneElement.setProperty("Zone_Number",
existZoneElement.getProperty("Zone_Number"));
   zoneElement.apply();
}
```

4.1.10 Deleting a document element using "cmf_deleteElement" action

Task:

Delete all the warehouse “Places” that contain less then N “Part Links.”

Solution description:

Use the "cmf_getElement" action to get all the “Places” of the document. You can then loop through the places, deleting those with less than N part links.

Write a method that, given a warehouse document and a threshold (method parameter names `documentId` and `n` respectively), deletes the places that have too few part links.

Note: You need to have “can_get” permissions on the document to run the following code.

Method

JavaScript

```
function deletePlacesByPartLinks(documentId, n) {
   var innovator = aras.newIOMInnovator();

   var placeElement = innovator.newItem("Place", "cmf_getElement");
   placeElement.setProperty("document_id", documentId);
   var partLinkElement = innovator.newItem("Part Link");
   placeElement.addRelationship(partLinkElement);
   // find all "Places" with there "PartLinks"
   var response = placeElement.apply();
   if (!response.isError()) {
      for (var i = 0; i < response.getItemCount() ; i++) {
         var place = response.getItemByIndex(i);
         var partLinkRelationships = place.getRelationships();
         var partLinkCount = partLinkRelationships.getItemCount();
         if (partLinkCount < n) {
            // remove place using "cmf_deleteElement"
            place.setAction("cmf_deleteElement");
            place.apply();
         }
      }
   } else {
      aras.AlertError(response.getErrorDetail());
      return;
   }
}
```

4.2 Configuring the CMF Document

To configure a CMF document, follow the procedures described in the following sections.

4.2.1 Creating an ItemType «Parts Warehouse»

1. Login to Aras innovator and select Administration **-> ItemTypes** from the TOC.
2. Create a new ItemType with the name **Parts Warehouse**. Give "TOC Access" for this item type, set appropriate "Permissions", set "Can Add" identity.
3. Also add new property "name" for this item type (type: string, length: 32, required: true).

4.2.2 Creating a CMF Content Type

1. Go to **Administration-> Content Modeling -> Content Types**. The following menu appears:

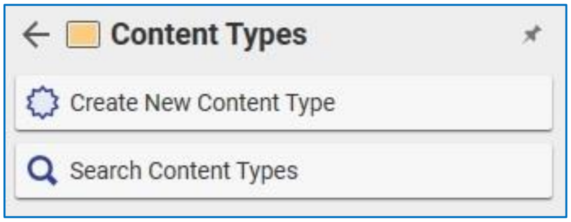


Figure 30.

2. Click **Create New Content Type**. The following window appears:

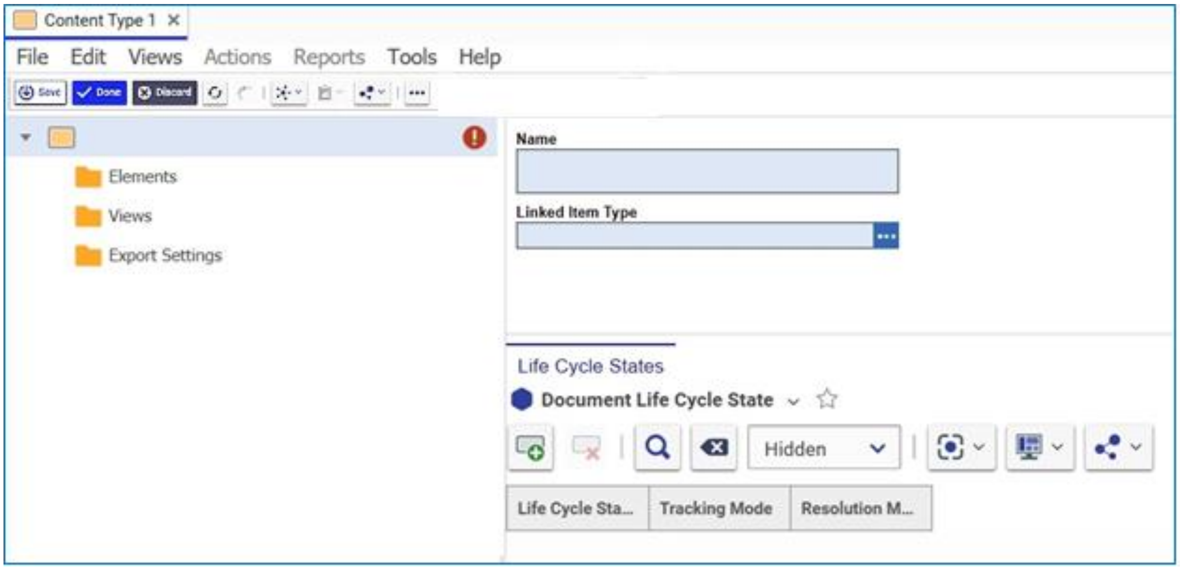
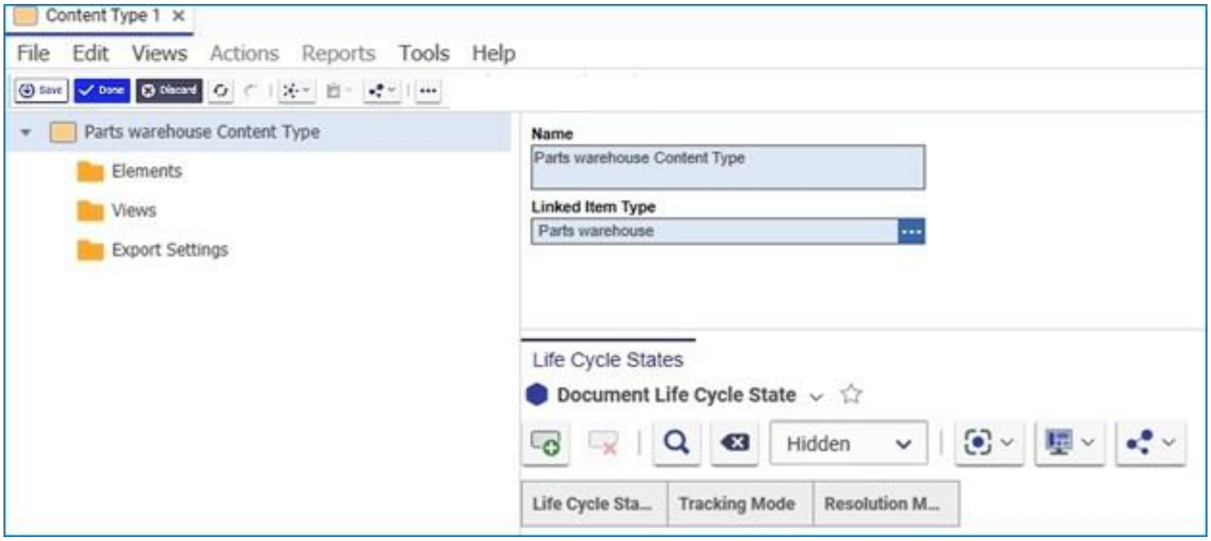


Figure 31.

3. Enter the name **Parts Warehouse Content Type** in the **Name** field.

4. Select **Parts Warehouse** as the Linked Item Type.



Name: Parts warehouse Content Type

Linked Item Type: Parts warehouse

Figure 32.

5. Save the Content Type.

4.2.3 Creating CMF Element Types and Properties

1. Add a new CMF Element by right clicking the **Elements** folder and selecting **Add Element Type** from the context menu.

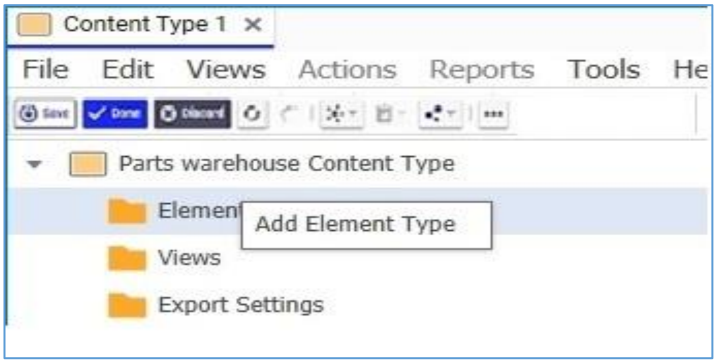
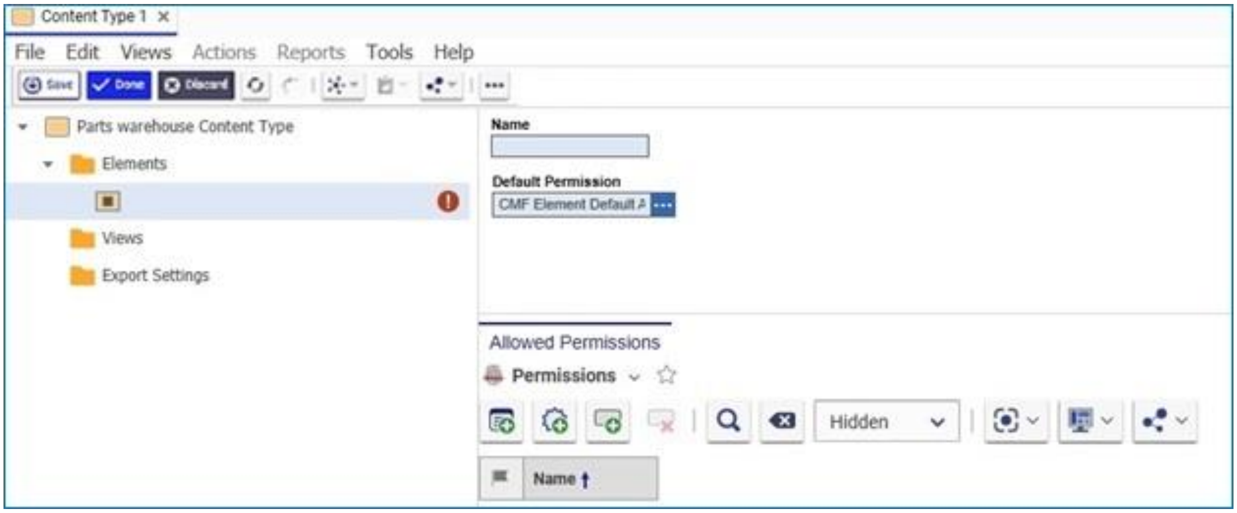


Figure 33.

The Element Type tab appears.



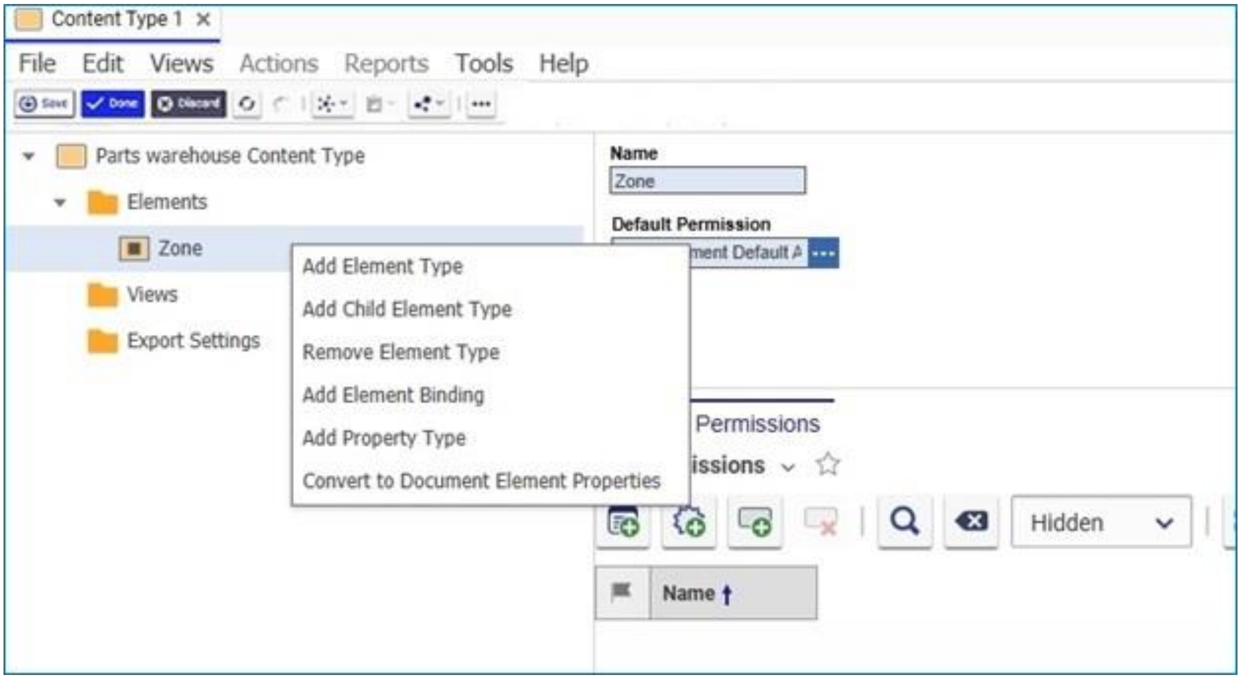
Name

Default Permission: CMF Element Default A

Figure 34.

2. Enter **Zone** in the Name field for the newly created Element.

3. Add a Property Type for this element by clicking the right mouse button on the Zone Element and selecting **Add Property Type** from the context menu.



Name: Zone

Default Permission: ment Default A

Figure 35.

4. The Property Type page appears.

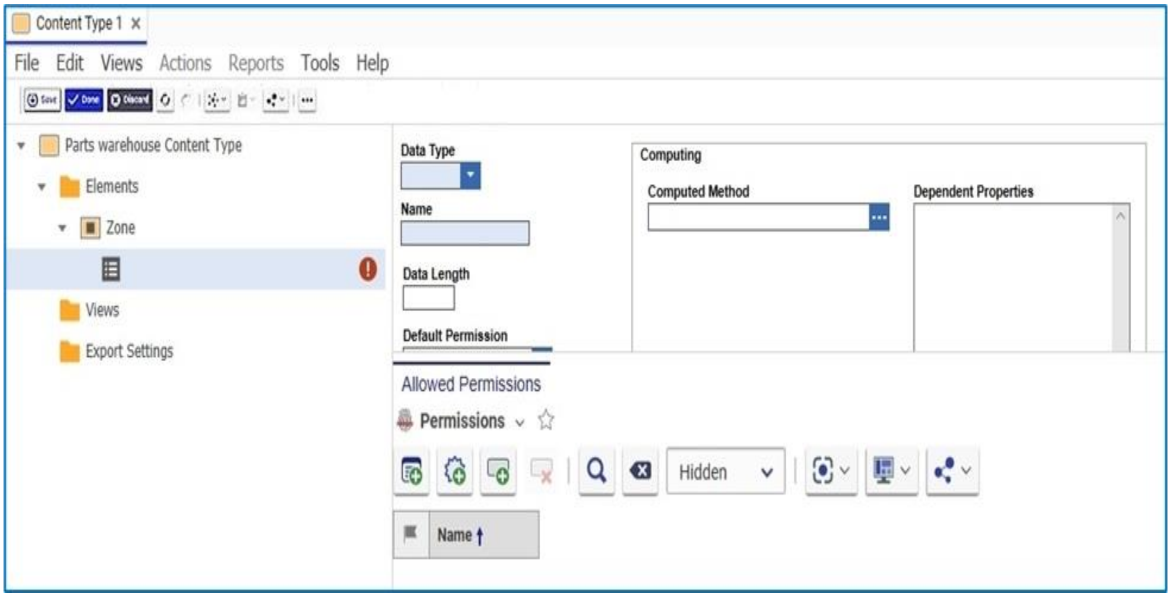
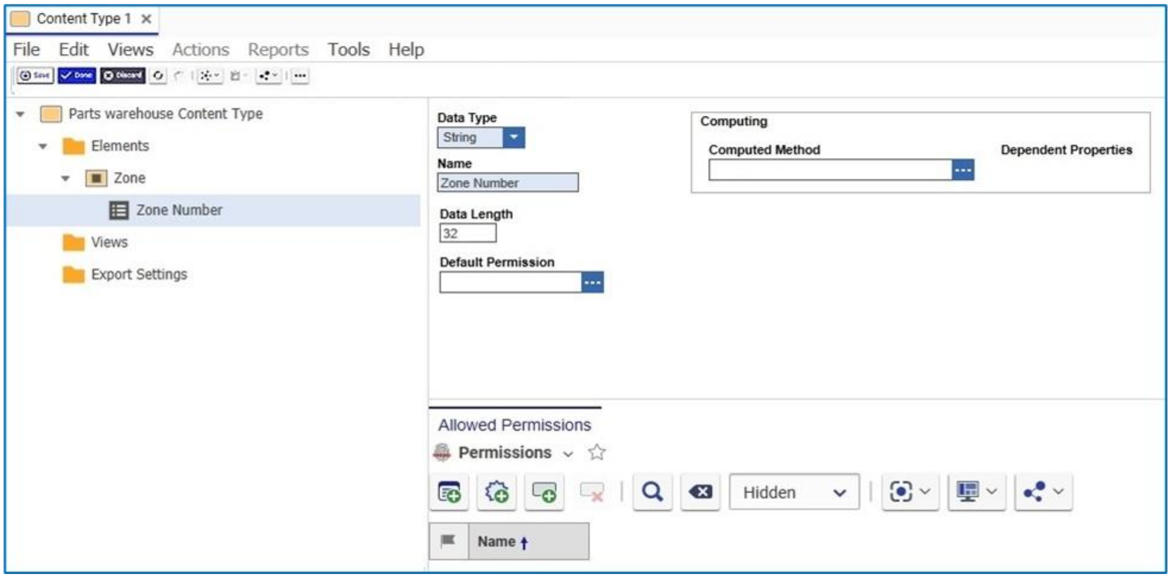


Figure 36.

5. Select **String** from the dropdown list in the Data Type field.

6. Enter **Zone Number** in the Name field.

7. Enter **32** in the Data Length field.



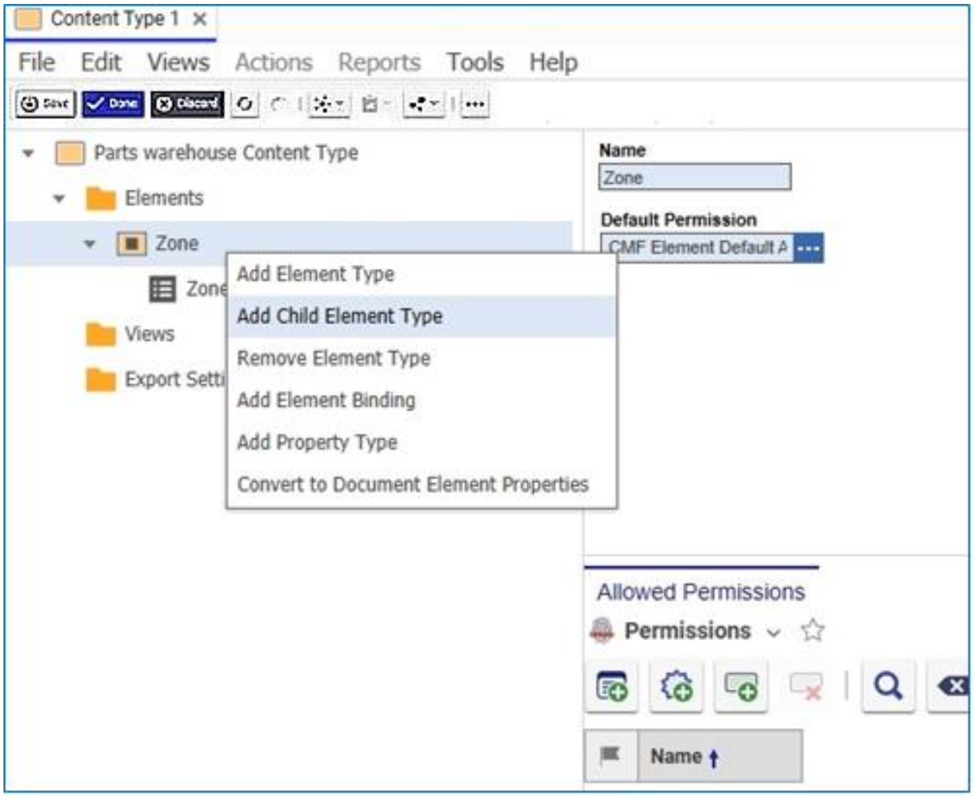
Data Type: String

Name: Zone Number

Data Length: 32

Figure 37.

8. Right click on the **Zone** element and select **Add Child Element Type** from the context menu.



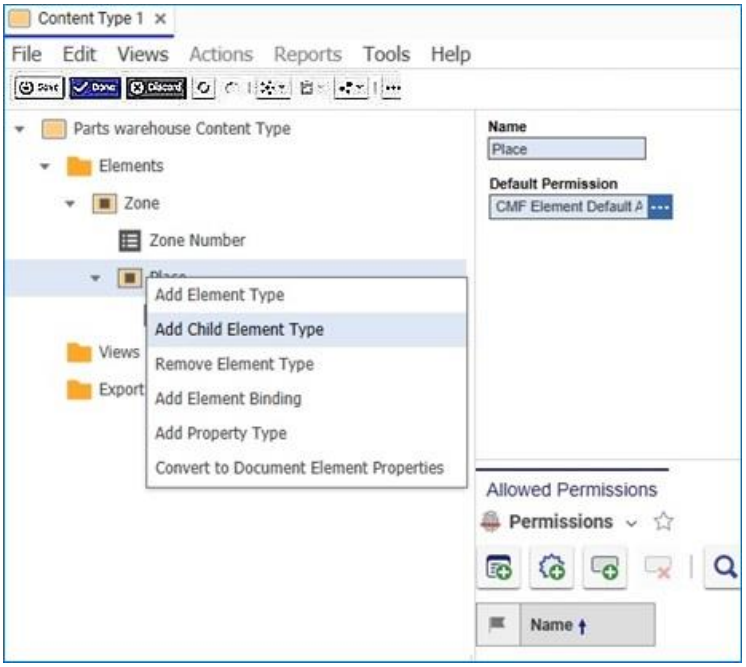
Name: Zone

Default Permission: CMF Element Default A

Figure 38.

9. Enter **Place** in the Name field for the Element Type and click **Save**.
10. Right click on the **Place** element and select **Add Property Type** from the context menu to create a new property type.
11. Select **String** in the Data Type field.
12. Enter **Place Number** in the Name field.
13. Enter **32** in the Data Length field.
14. Click **Save**.

15. Right click on the **Place** element to add a Child Element Type.

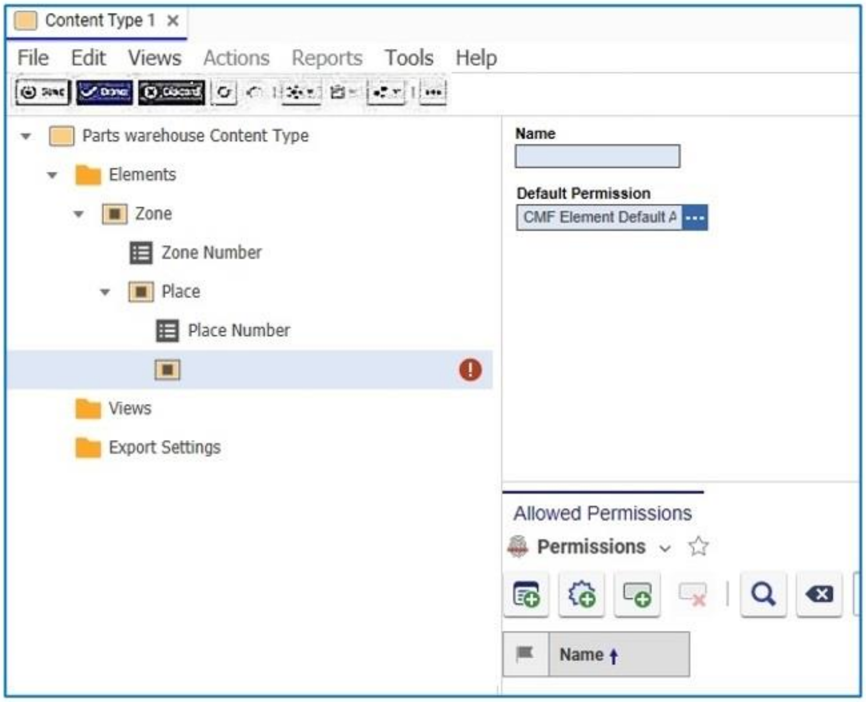


Name: Place

Default Permission: CMF Element Default A

Figure 39.

16. Select **Add Child Element Type** from the context menu. The Element Type page appears.



Name:

Default Permission: CMF Element Default A

Figure 40.

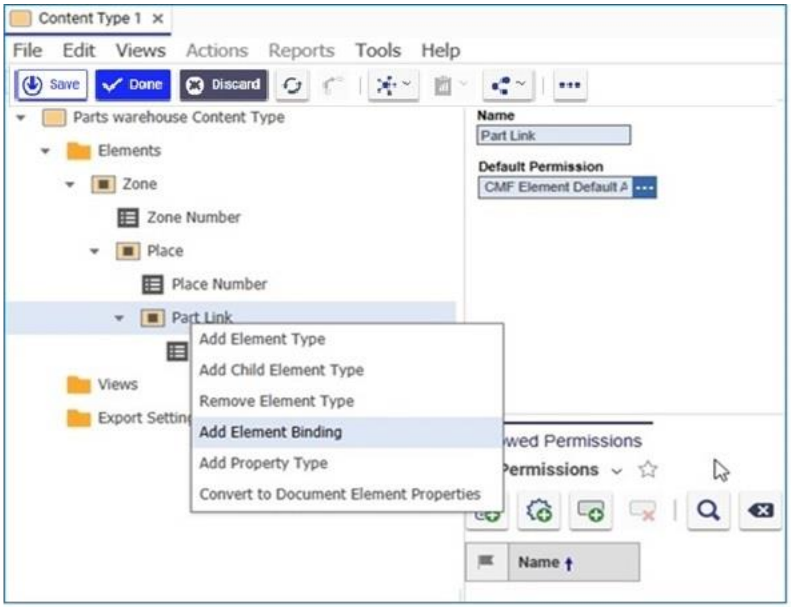
17. Enter **Part Link** in the Name field and click **Save**.

18. Right click on **Part Link** and select **Add Property Type** from the context menu.

19. Choose **String** from the dropdown list in the **Data Type** field.
20. Enter **32** in the Data Length field and click **Save**.

4.2.4 Adding Element Binding on a Document Item Type

1. Right click **Part Link** and select **Add Element Binding** from the context menu.

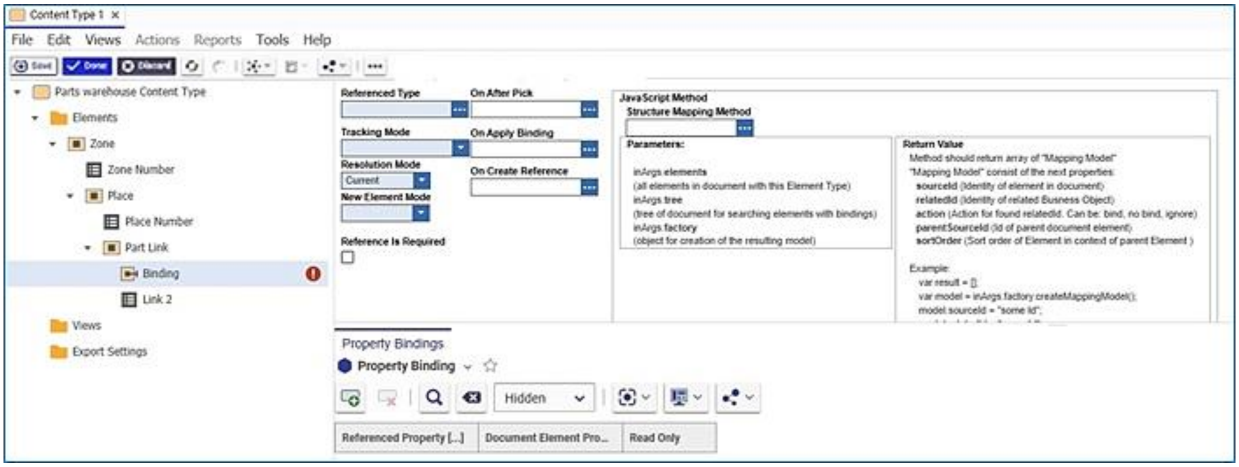


Name: Part Link

Default Permission: CMF Element Default A

Figure 41.

The Binding element appears, as shown in Figure 42.



Resolution Mode: Current

Reference Is Required: □

Parameters:

inArgs.elements
(all elements in document with this Element Type)
inArgs.tree
(tree of document for searching elements with bindings)
inArgs.factory
(object for creation of the resulting model)

Return Value

Method should return array of "Mapping Model"
"Mapping Model" consist of the next properties:
sourceId (Identity of element in document)
relatedId (Identity of related Business Object)
action (Action for found relatedId. Can be: bind, no bind, ignore)
parentSourceId (Id of parent document element)
sortOrder (Sort order of Element in context of parent Element)

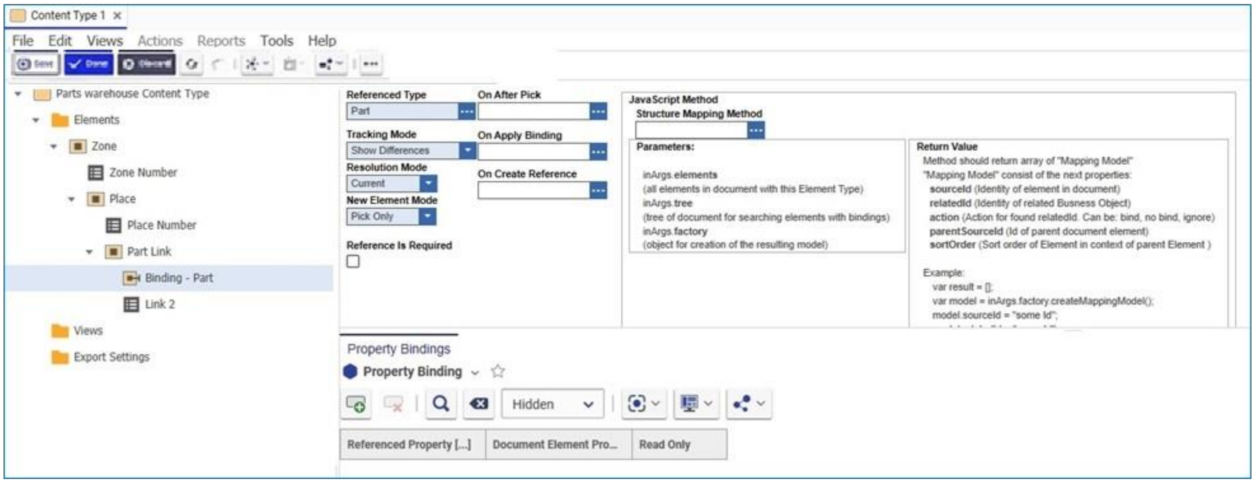
Example:

```
var result = [];
var model = inArgs.factory.createMappingModel();
model.sourceId = "some Id";
```

Figure 42.

2. Select **Part** as a Referenced Type property.
3. Select **Show Differences** from the Tracking Mode dropdown list.
4. Select **Current** from the Resolution Mode dropdown list.
5. Select the **Reference is required** checkbox and click **Save**.
6. Select **Pick Only** from the **New Element Mode** dropdown list.

7. Click the **New Property Binding** icon on the Property Bindings tab to add a Property Bindings relationship for the Element Binding.



Referenced Type: Part
Tracking Mode: Show Differences
Resolution Mode: Current
New Element Mode: Pick Only
Reference Is Required ☐
On After Pick
On Apply Binding
On Create Reference

JavaScript Method
Structure Mapping Method

Parameters:

inArgs.elements
(all elements in document with this Element Type)
inArgs.tree
(tree of document for searching elements with bindings)
inArgs.factory
(object for creation of the resulting model)

Return Value
Method should return array of "Mapping Model"
"Mapping Model" consist of the next properties:
sourceId (Identity of element in document)
relatedId (Identity of related Busness Object)
action (Action for found relatedId. Can be: bind, no bind, ignore)
parentSourceId (Id of parent document element)
sortOrder (Sort order of Element in context of parent Element)

Example:
```
var result = [];
var model = inArgs.factory.createMappingModel();
model.sourceId = "some Id";
```

Figure 43.

8. Click the New Property Binding icon and click the ⋯ in the Referenced Property column. The Select Items Properties dialog appears.

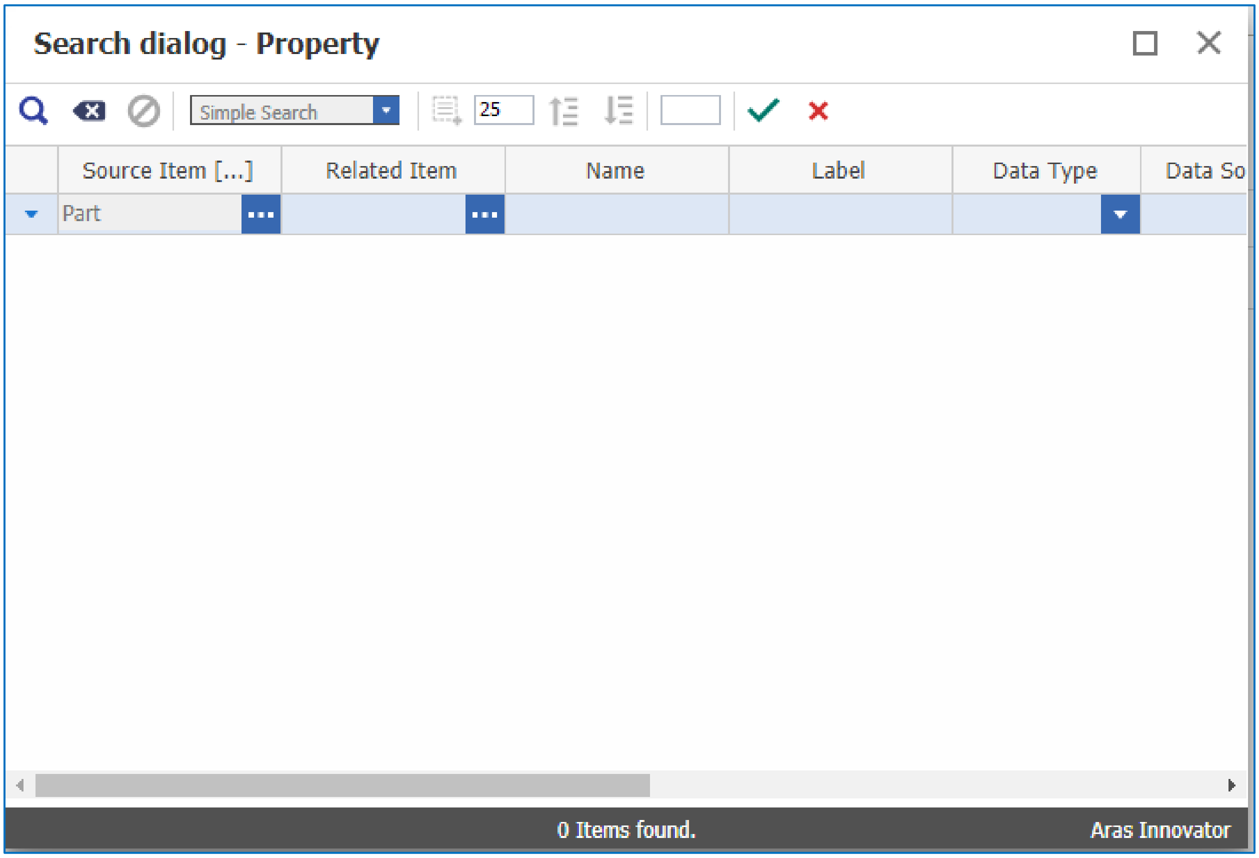
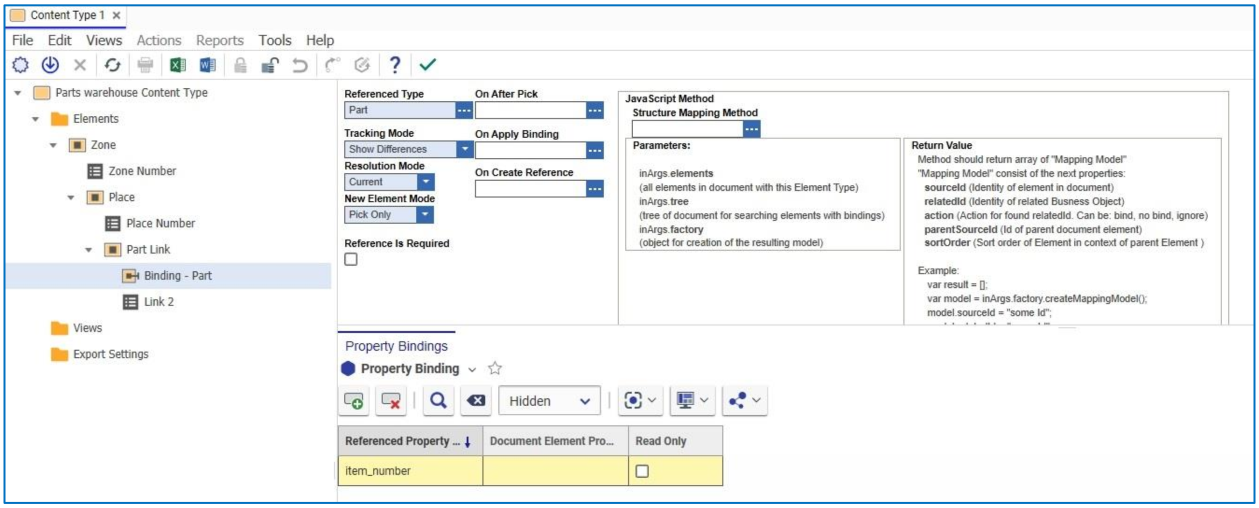


Figure 44.

9. Click the Run Search icon and select **item_number** from the property list.

10. Click the **Document Element Property** column and click the ··· to access the Search dialog.
11. Enter **Part Number** in the Name column and click the Search icon.
12. Select **Part Number** and click the ✓ icon to select it.



Referenced Type: Part
Tracking Mode: Show Differences
Resolution Mode: Current
New Element Mode: Pick Only
Reference Is Required: □
On After Pick
On Apply Binding
On Create Reference

JavaScript Method
Structure Mapping Method

Parameters:

inArgs.elements
(all elements in document with this Element Type)
inArgs.tree
(tree of document for searching elements with bindings)
inArgs.factory
(object for creation of the resulting model)

Return Value
Method should return array of "Mapping Model"
"Mapping Model" consist of the next properties:
sourceId (Identity of element in document)
relatedId (Identity of related Busness Object)
action (Action for found relatedId. Can be: bind, no bind, ignore)
parentSourceId (Id of parent document element)
sortOrder (Sort order of Element in context of parent Element)

Example:
```
var result = [];
var model = inArgs.factory.createMappingModel();
model.sourceId = "some Id";
```

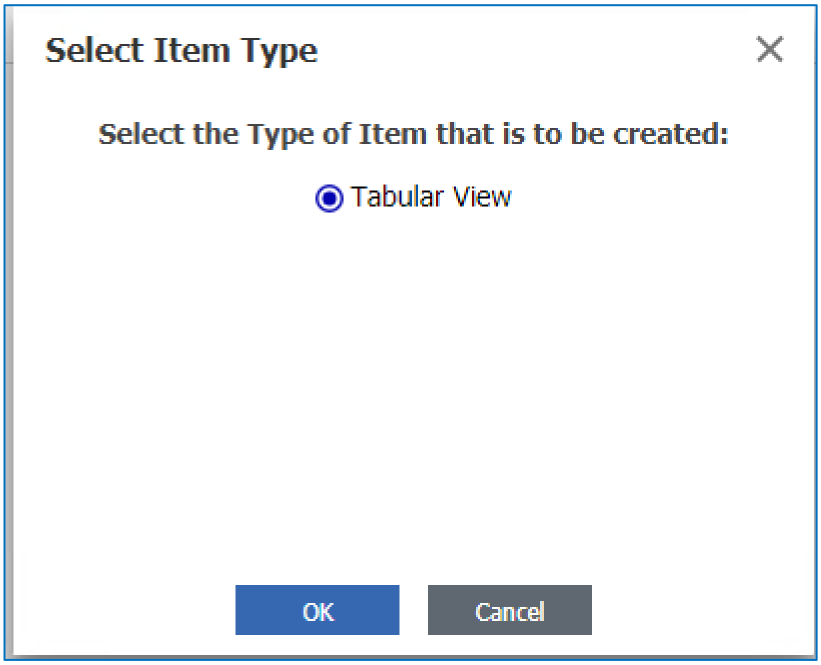
Referenced Property ...	Document Element Pro...	Read Only
item_number		□

Figure 45.

4.2.5 Adding a View for a Content Type

Use the following procedure to add a new "View" for the Parts Warehouse Content Type:

1. Right click the **View** folder and select **Add View** from the context menu. The Select ItemType dialog appears:



Select Item Type

Select the Type of Item that is to be created:

● Tabular View

Figure 46.

2. Click **OK**. The Tabular View page appears.

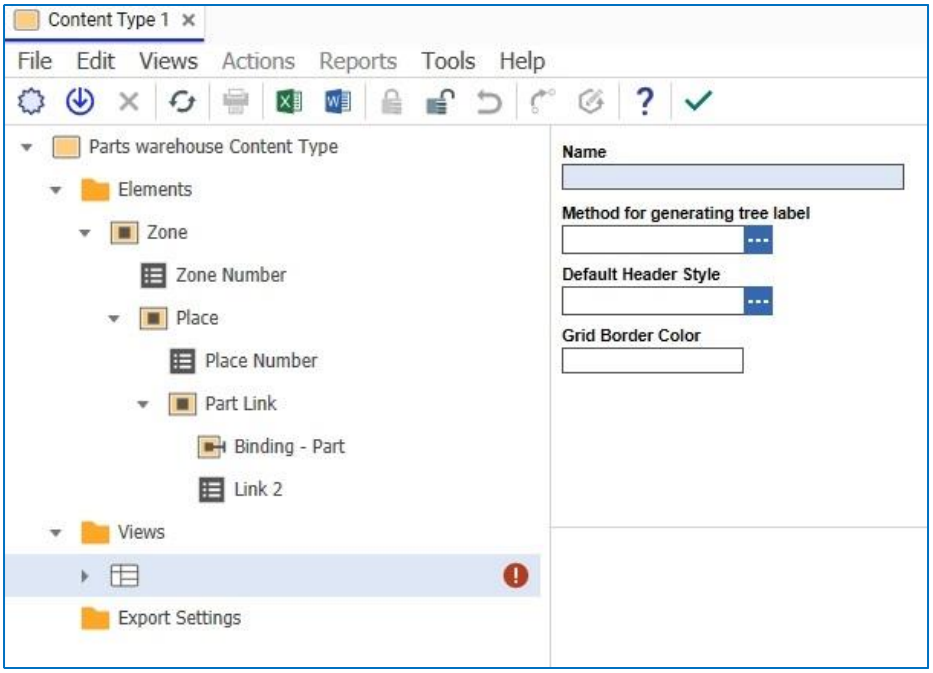
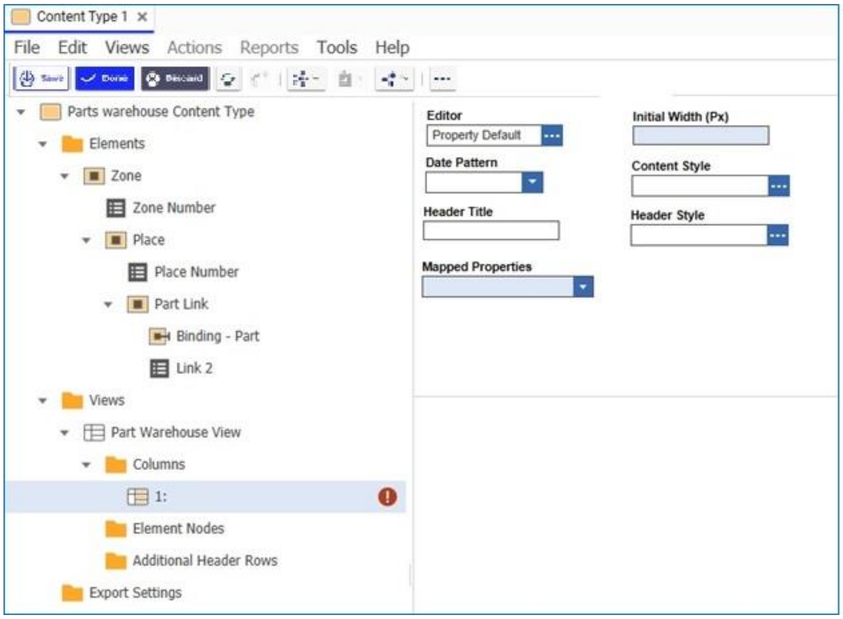


Figure 47.

3. Enter **Part Warehouse View** in the Name field and click **Save**.
4. Expand the Part Warehouse View tree and right-click **Columns**.
5. Select **Add Column** from the context menu. The tabular View Column window appears.

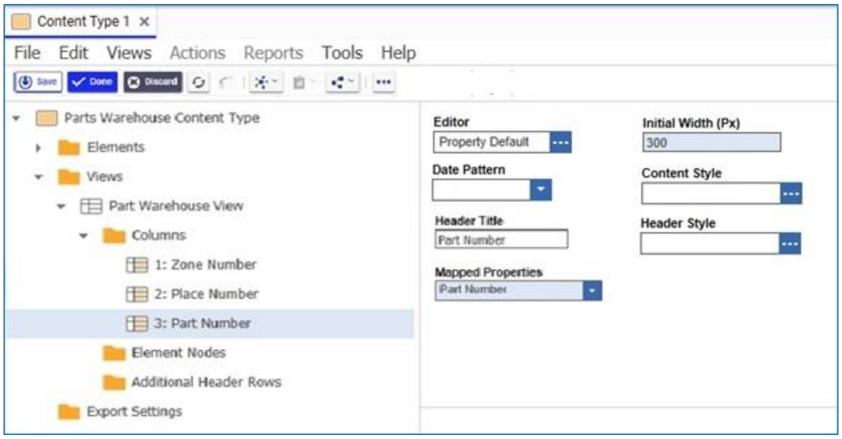


Editor: Property Default

Figure 48.

6. Enter **Zone Number** in the Header Title field.
7. Select **Zone Number** from the dropdown list in the Mapped Properties field.
8. Enter **300** in the Initial Width field and click **Save**.

9. Right click 1: Zone Number and select **Add Column** from the context menu to add another tabular view.
10. Enter Place Number in the **Header Title** and **Mapped Properties** fields.
11. Enter 300 in the **Initial Width** field and click **Save**.
12. Right click on the Place Number column and select **Add column** from the context menu.
13. Enter **Part Number** in the Header Title and Mapped Property fields.
14. Enter **300** in the Initial Width field and click **Save**.



Editor: Property Default
Initial Width (Px): 300
Date Pattern:
Content Style:
Header Title: Part Number
Header Style:
Mapped Properties: Part Number

Figure 49.

15. Save, Unclaim and close **Parts Warehouse Content** Type.

4.2.6 Adding an "OnFormPopulate" Event and Name Property

1. Select **Administration-> Forms**. The following menu appears:

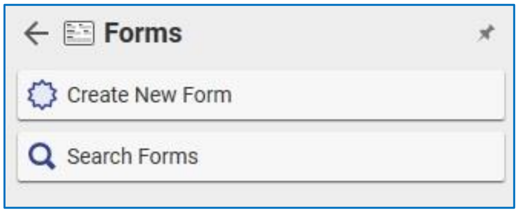


Figure 50.

2. Click **Search Forms**. The Form Search grid appears.

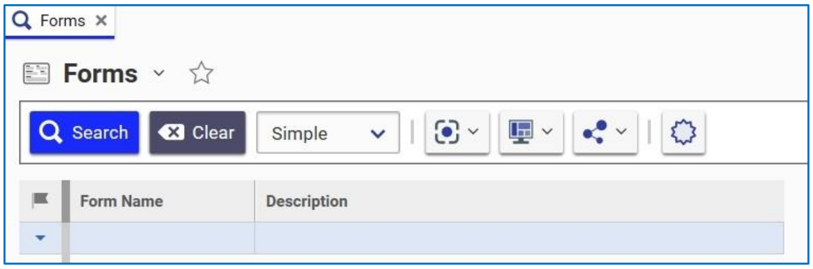
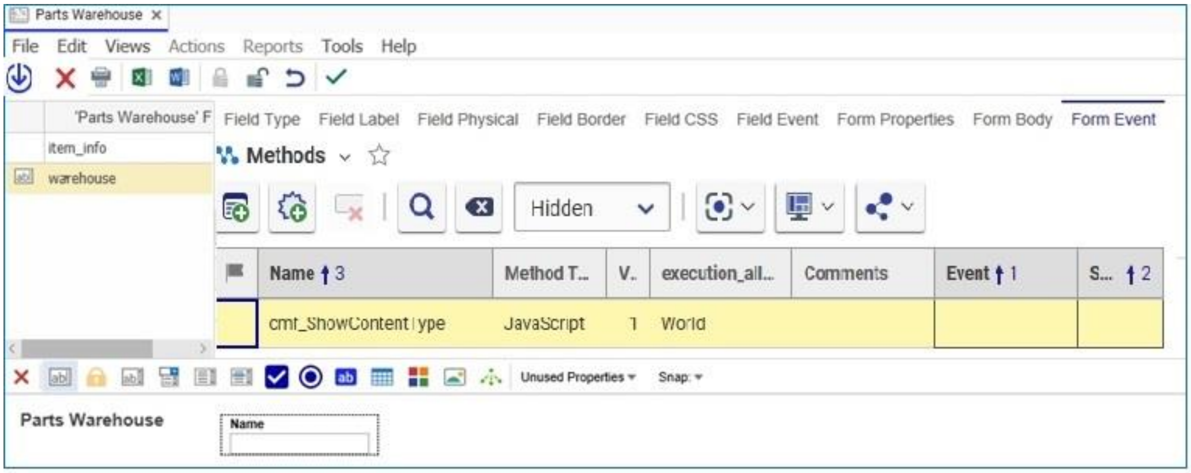


Figure 51.

3. Open the Parts Warehouse form and click 🔒 to claim the form for editing.

4. Select the Form Event tab and click the **Add Methods** icon . The Search dialog box appears.
5. Search for and select the **cmf_ShowContentType** method. It appears in the Name column.
6. Click the cell in the Event column and select the **onFormPopulated** event.
7. Add the **Name** property to the form (if it does not already exist).


Parts Warehouse
File Edit Views Actions Reports Tools Help
'Parts Warehouse' F
item_info
warehouse
Field Type Field Label Field Physical Field Border Field CSS Field Event Form Properties Form Body Form Event
Methods
Hidden
Unused Properties
Snap:
Parts Warehouse
Name

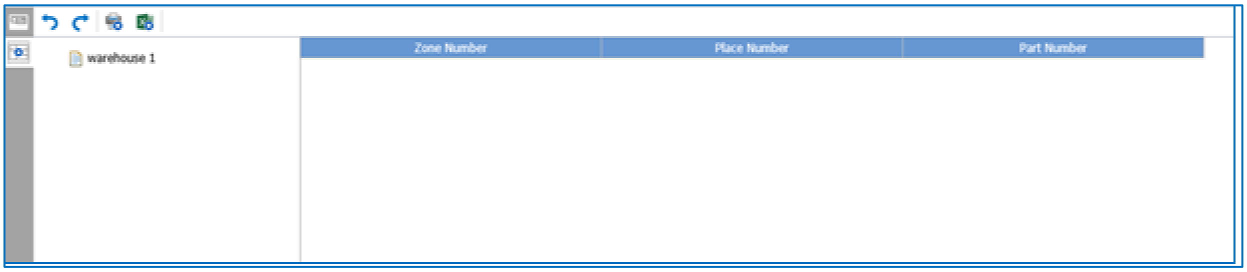
	Name ↑3	Method T...	V..	execution_all...	Comments	Event ↑1	S... ↑2
	cmf_ShowContentType	JavaScript	1	World			

Figure 52.

8. Save, Unclaim and Close the Form.

4.2.7 Checking the Metadata Configuration

1. Go to **Administration>ItemTypes** in the TOC and select **Search ItemTypes**. The ItemTypes search grid appears.
2. Select **Parts Warehouse** and click **Edit**.
3. Click the **Views** tab, select **Create Related** from the dropdown list. Click the **Create Item** icon to add a view named “Warehouse 1” and save.
4. Then go to “View” using the sidebar button. You should see the result of metadata configuration.



Zone Number	Place Number	Part Number

Figure 53.

Configuration is completed.