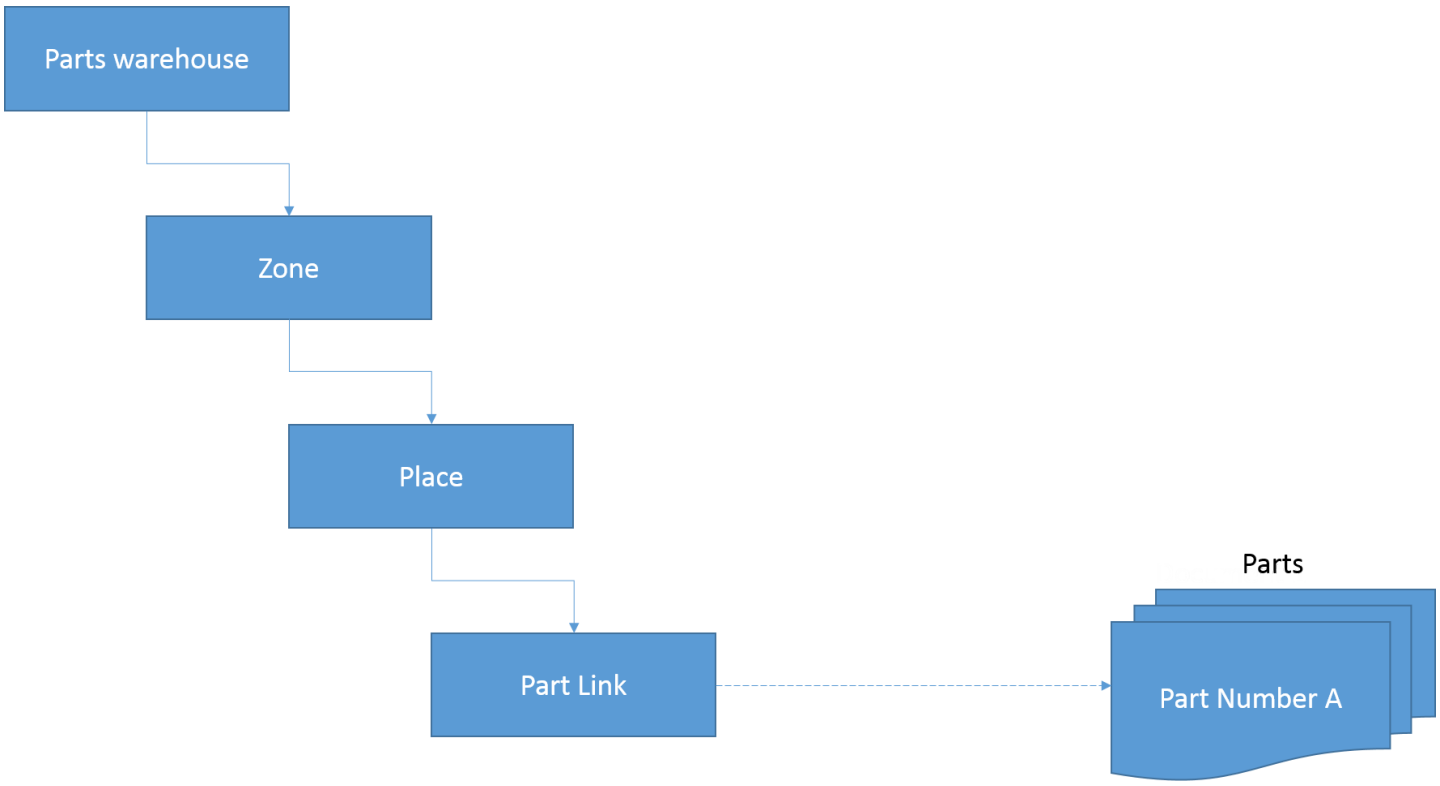
Content Modeling Framework (CMF) API

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.

Sample CMF Document Parts Warehouse

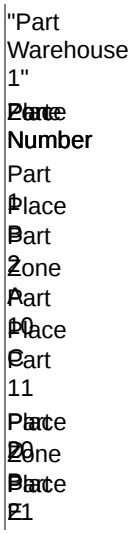
Our sample document describes an imaginary warehouse that is partitioned into Zones, each of which is subdivided into Places. Each Place can contain one or more Parts items. The following data model diagram illustrates the structure of the document.



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

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.



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.”

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

> **Important**
>
> 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();
```

```
}
}
}
```

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*) .

> **Important**
>
> 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 };
}
}
```

Reading a Document Element Using the cmf_getElement action

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.

> **Important**
>
> 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();
}
}
```

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.

> **Important**
>
> 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 [];
}
}
```

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.

> **Important**
>
> 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 [];
}
}
```

Updating a Document Element Using the cmf_updateElement Action

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	Date After

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

Important

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();
}
}
}
}
```

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:

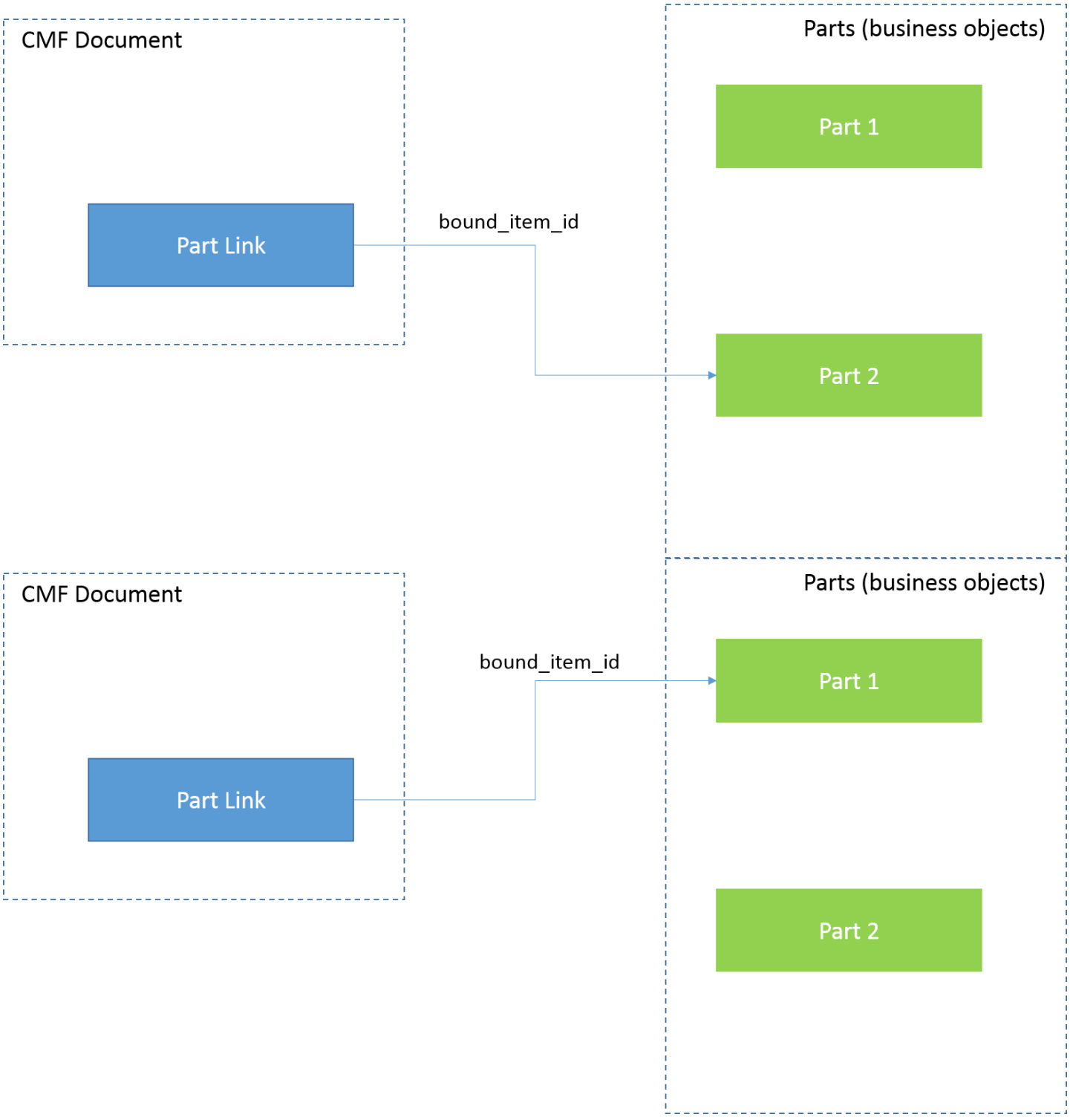
Table 1: The document identifier

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.



Before: After:

Important

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();
}
```

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.

Important

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).

Important

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();
}
```

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.

> **Important**
>
> 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;
}
}
```

Configuring the CMF Document

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

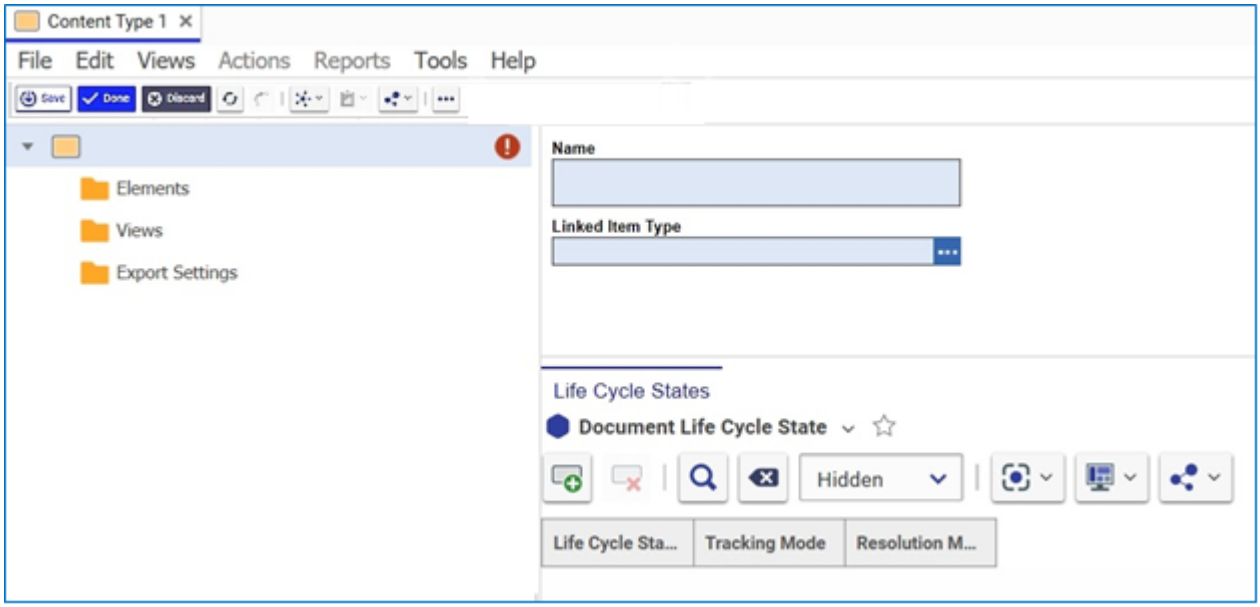
Creating an ItemType «Parts Warehouse»

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).

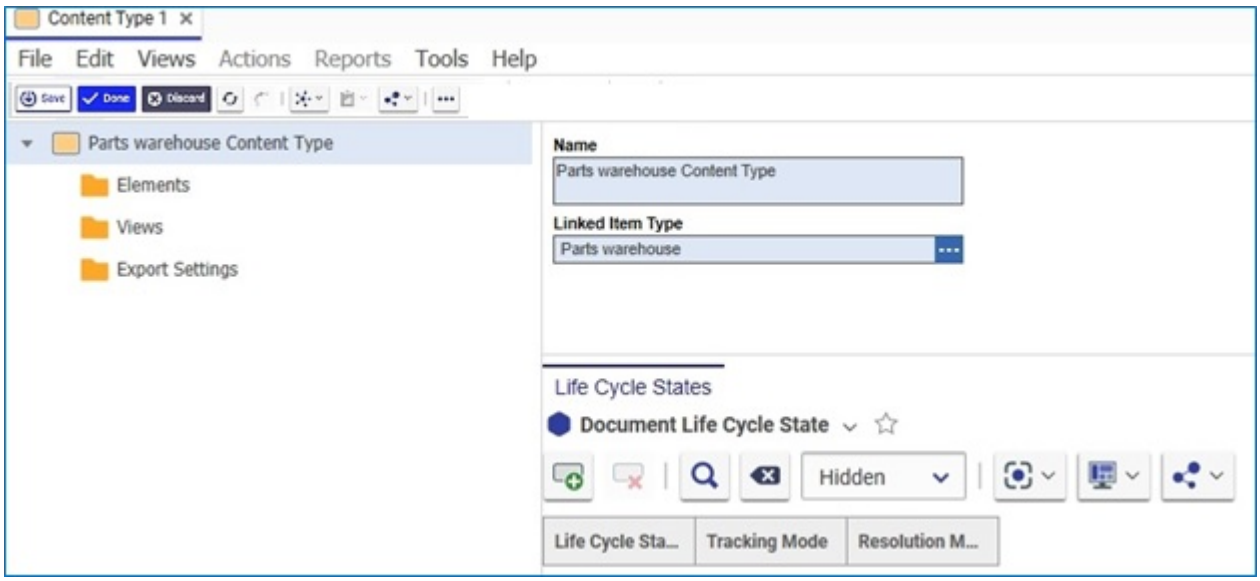
Creating a CMF Content Type

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

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



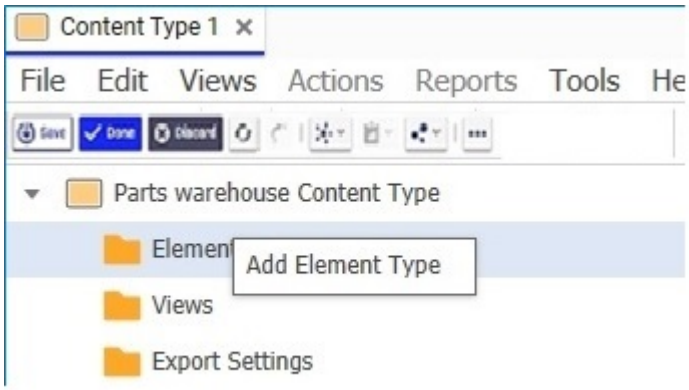
3. Enter the name **Parts Warehouse Content Type**in the **Name** field.
4. Select **Parts Warehouse** as the Linked Item Type.



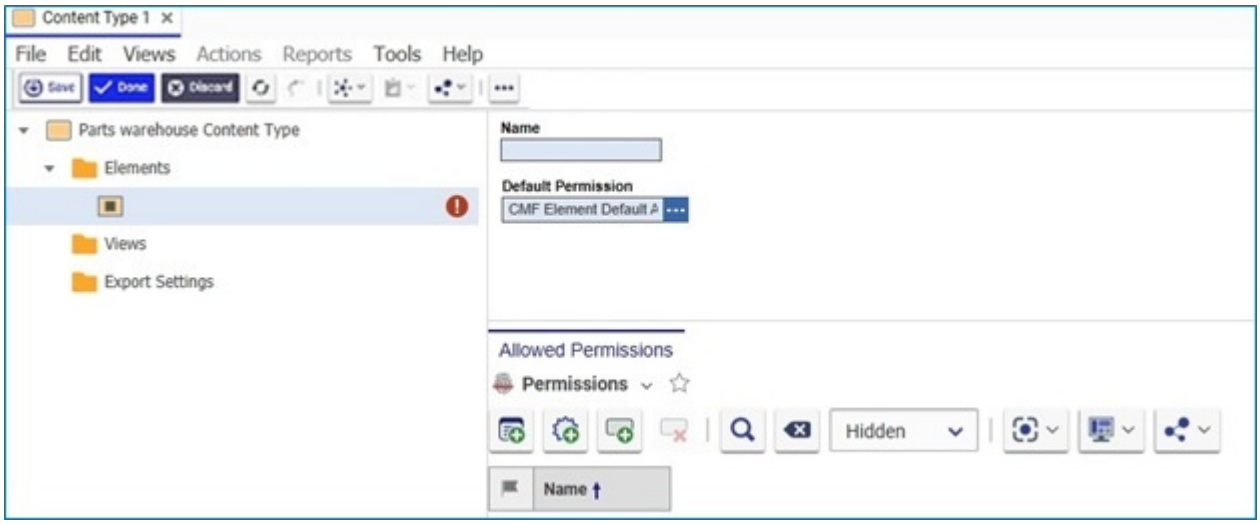
5. Save the Content Type.

Creating CMF Element Types and Properties

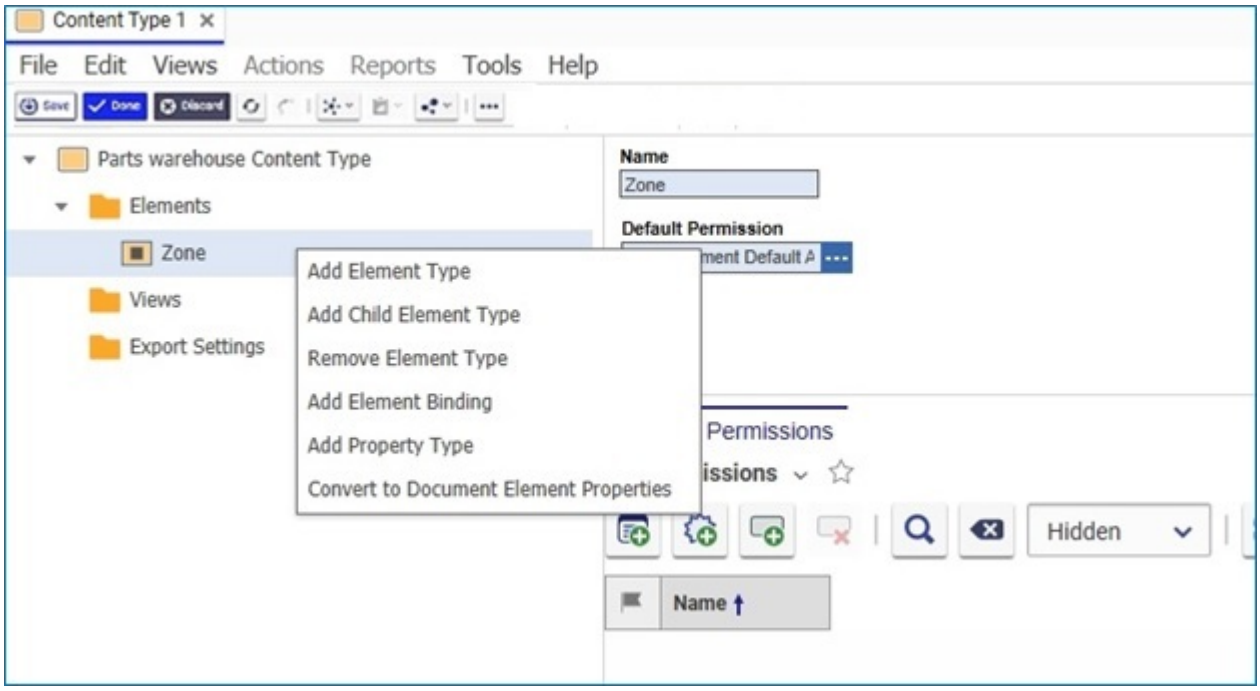
1. Add a new CMF Element by right clicking the **Elements** folder and selecting **Add Element Type** from the context menu.



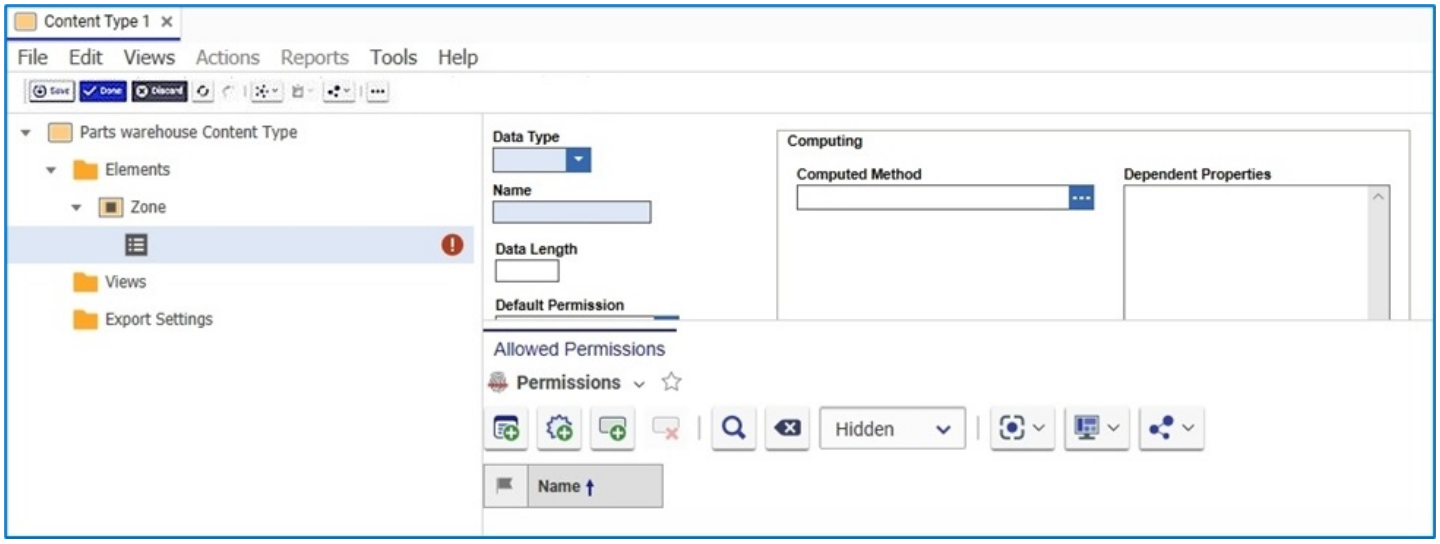
The Element Type tab appears.



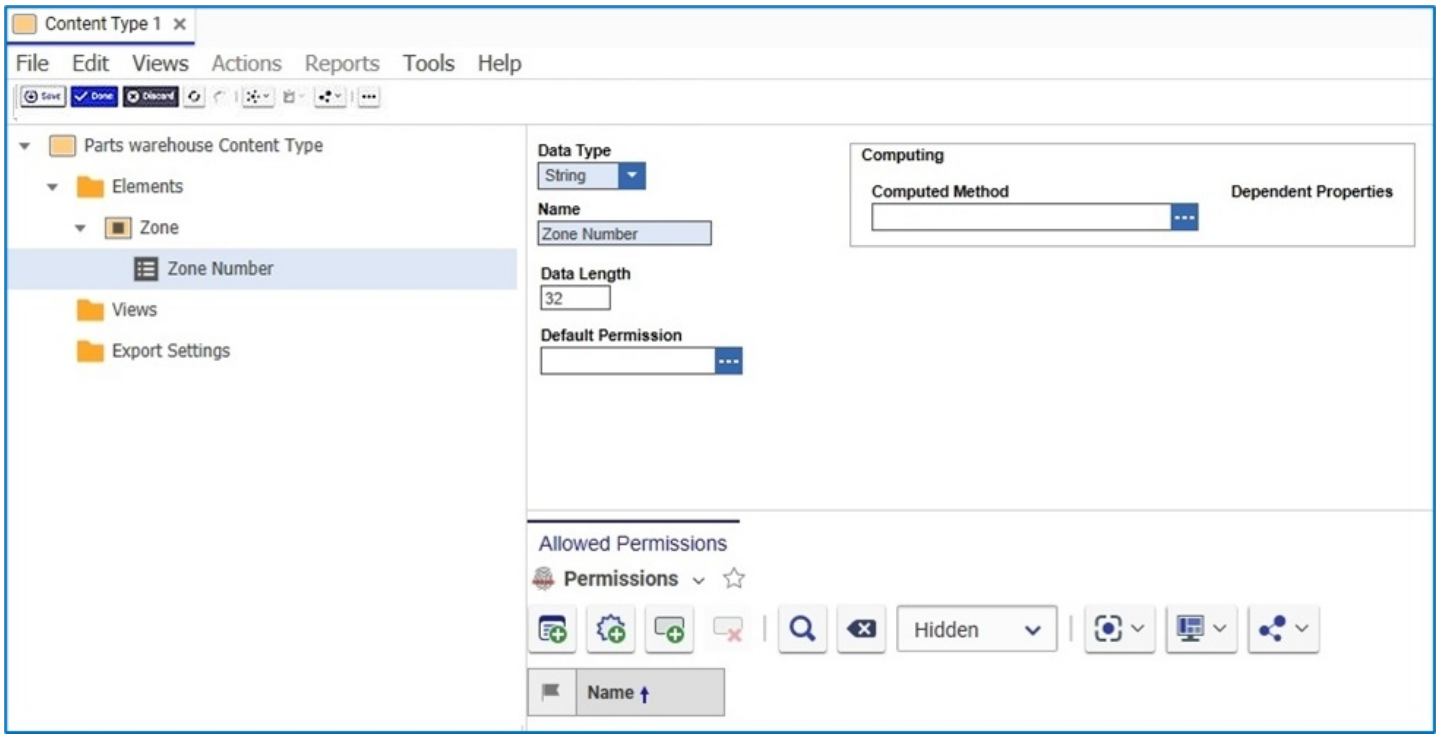
2. Enter **Zone** in the Name field for the newly created Element.
3. Add a Property Type for this element by clicking the right mouse button on the Zone Element and selecting **Add Property Type** from the context menu.



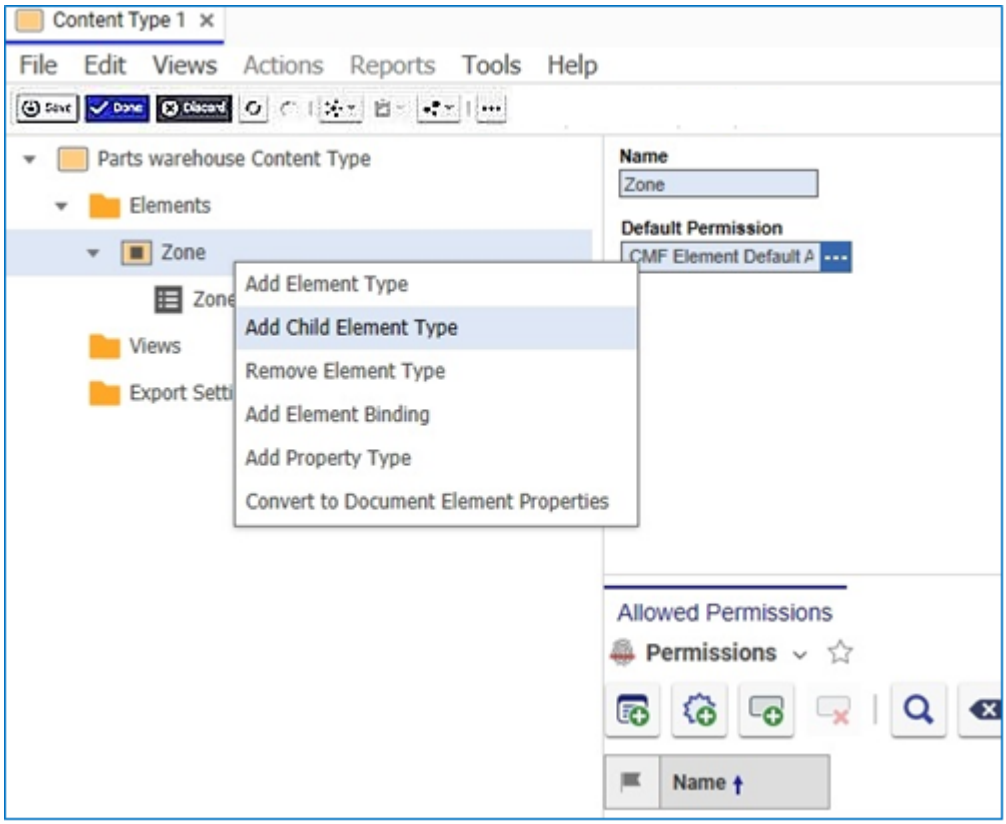
4. The Property Type page appears.



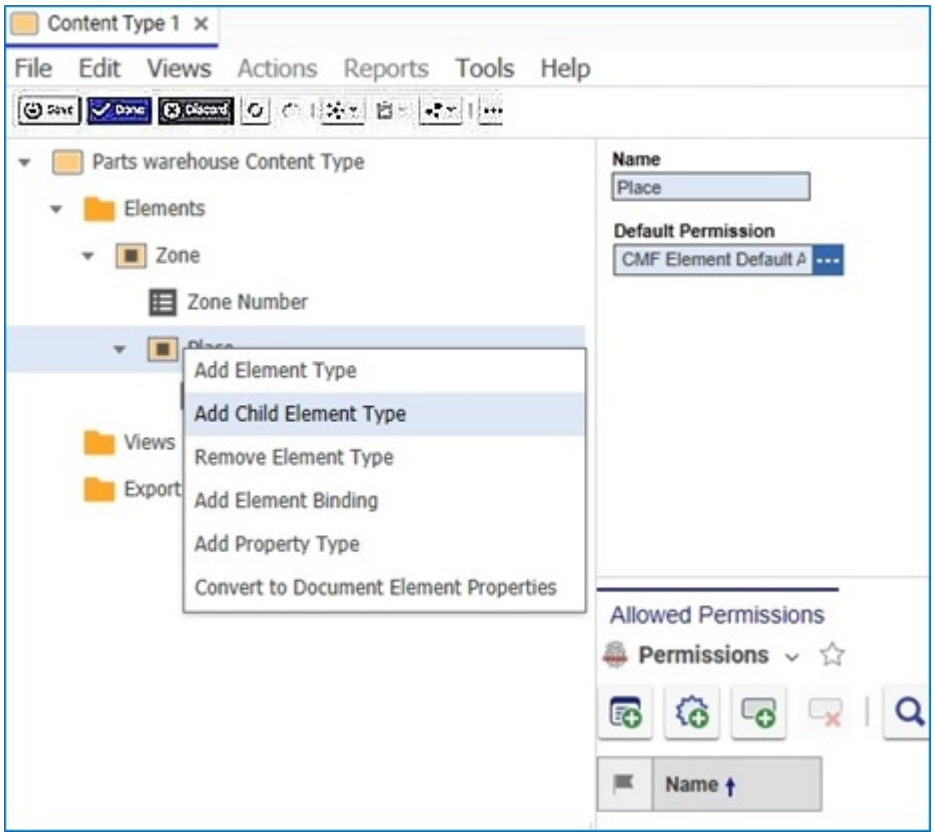
5. Select **String** from the dropdown list in the Data Type field.
6. Enter **Zone Number** in the Name field.
7. Enter **32** in the Data Length field.



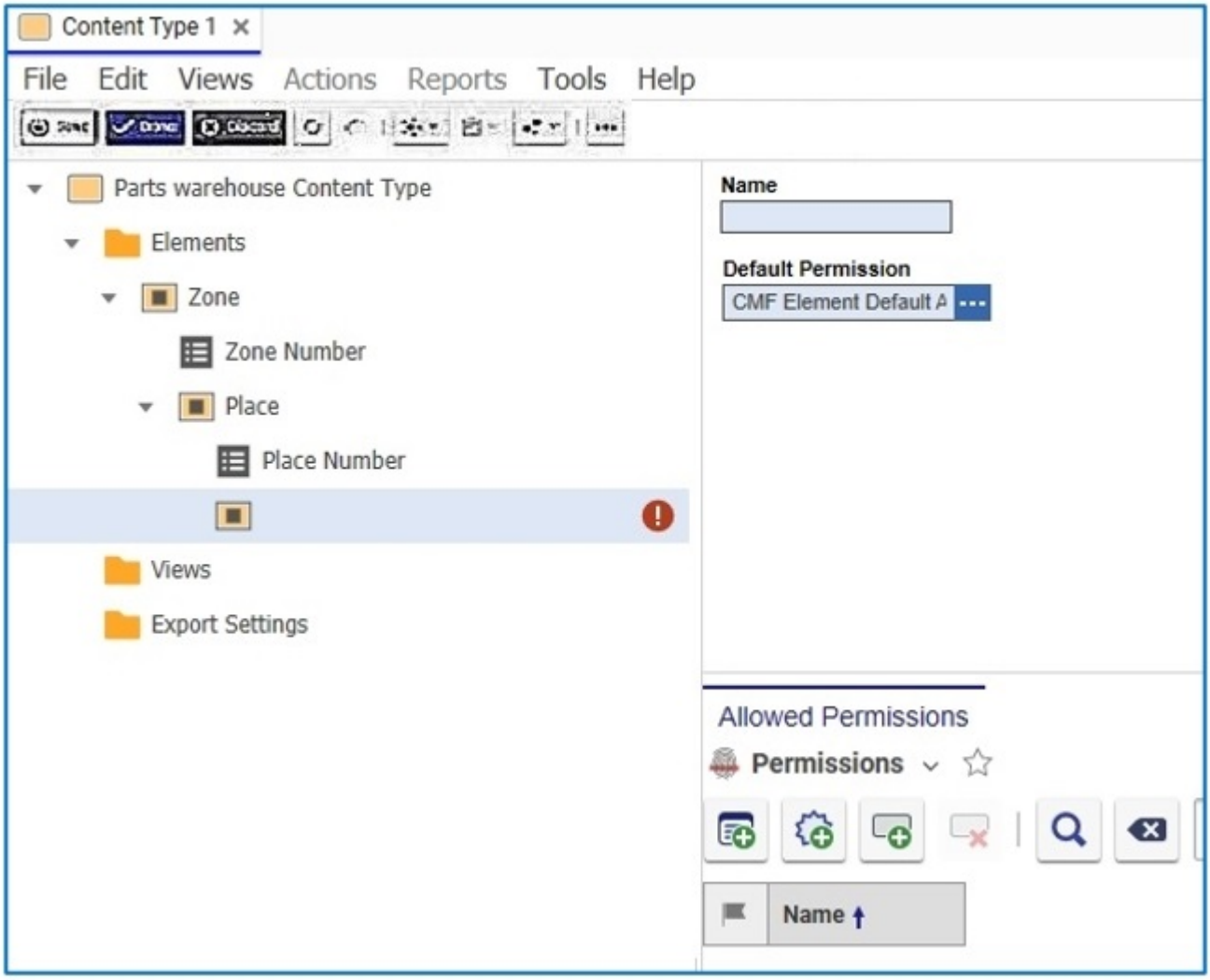
8. Right click on the **Zone** element and select **Add Child Element Type** from the context menu.



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.



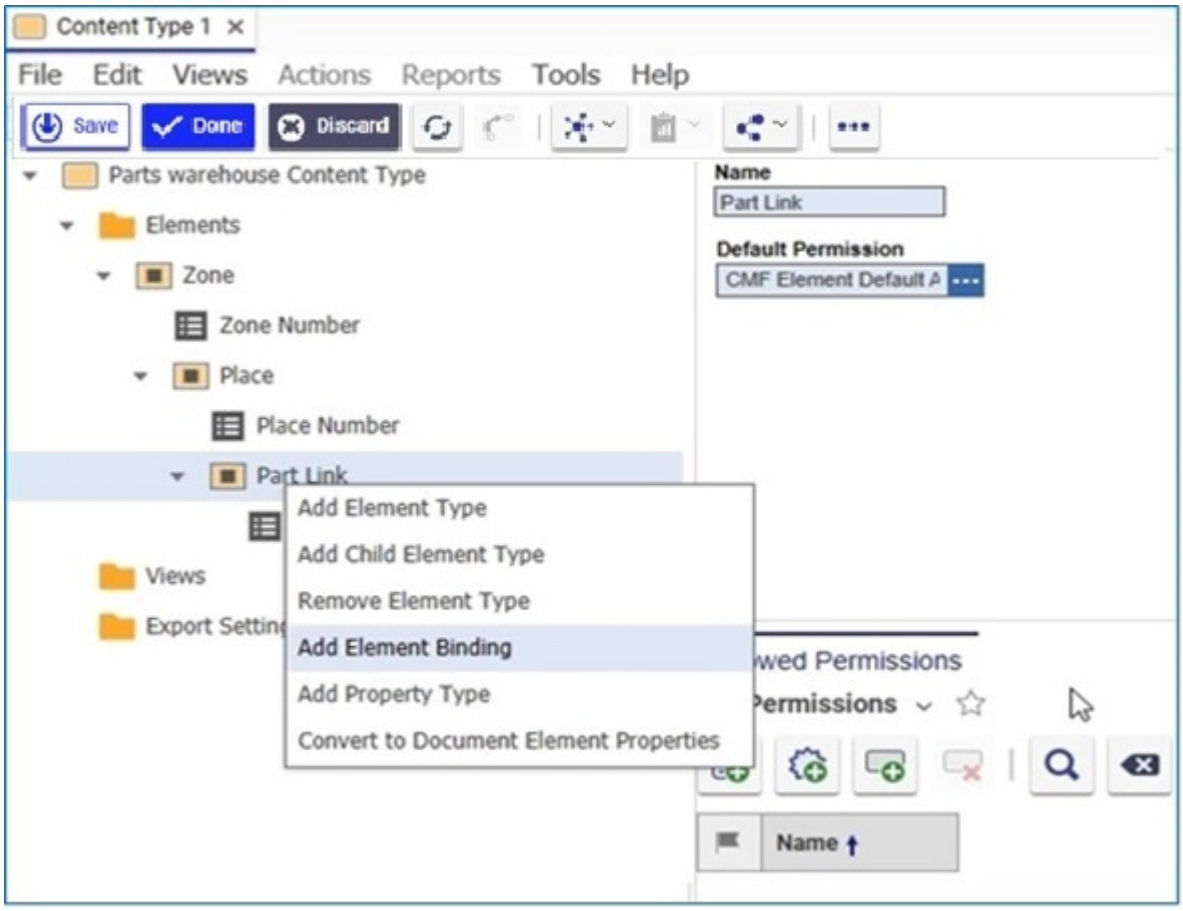
16. Select **Add Child Element Type** from the context menu. The Element Type page appears.



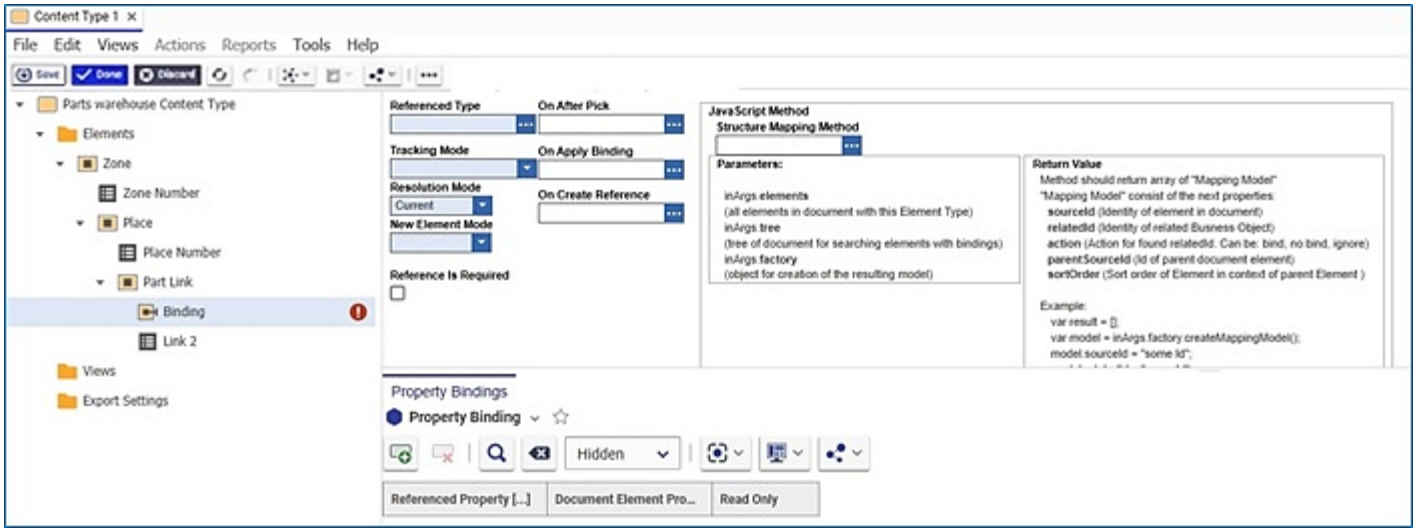
17. Enter **Part Link** in the Name field and click **Save**.
18. Right click on **Part Link** and select **Add Property Type** from the context menu.
19. Choose **String** from the dropdown list in the **Data Type** field.
20. Enter **32** in the Data Length field and click **Save**.

Adding Element Binding on a Document Item Type

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

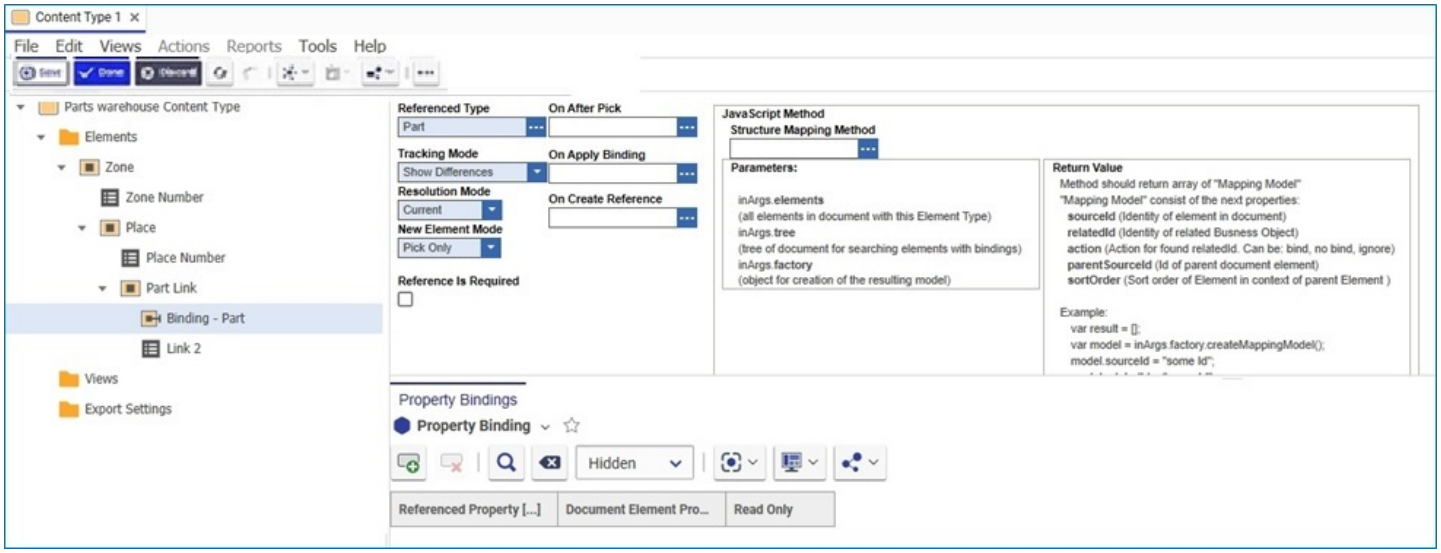


The Binding element appears, as shown in Figure 42.

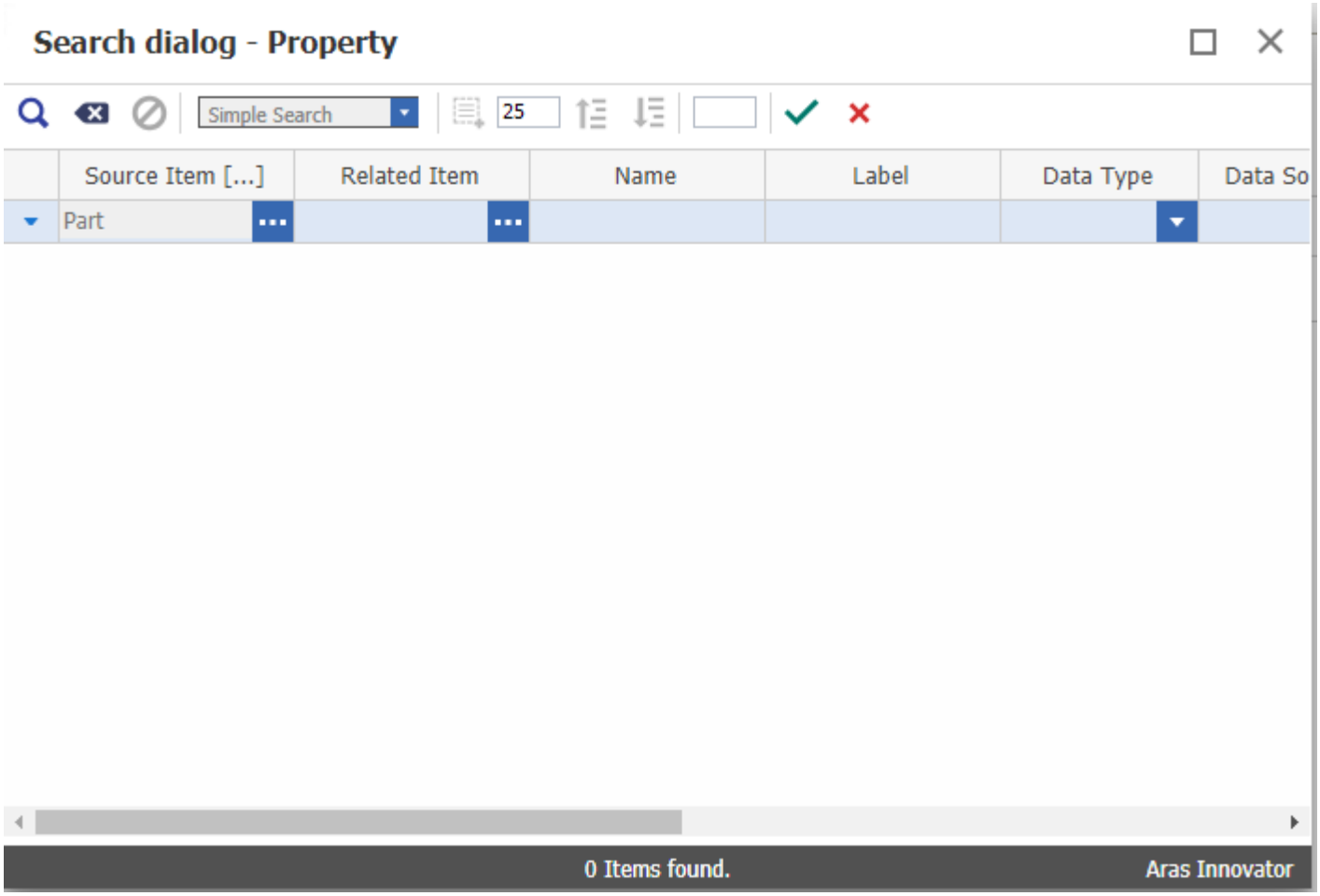


2. Select **Part** as a Referenced Type property.
3. Select **Show Differences** from the Tracking Mode dropdown list.
4. Select **Current** from the Resolution Mode dropdown list.
5. Select the **Reference is required** checkbox and click **Save**.
6. Select **Pick Only** from the **New Element Mode**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

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



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

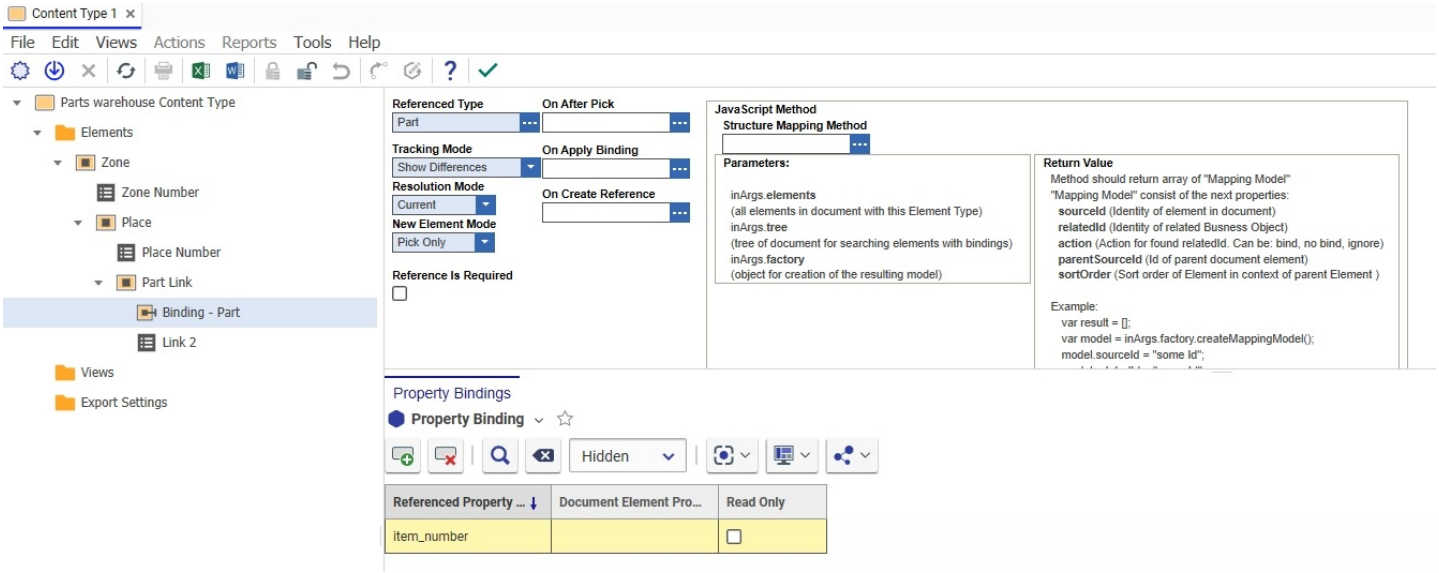


7. Click the Run Search icon and select **item_number** from the property list.

Click the Document Element Property column and click the null to access the Search dialog.

8. Enter **Part Number** in the Name column and click the Search icon.

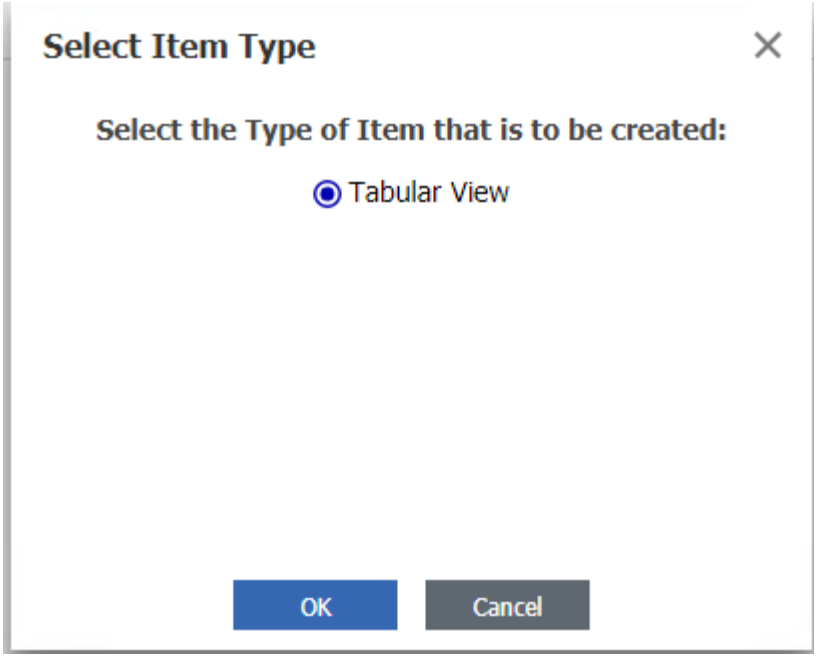
Select Part Number and click the null icon to select it.



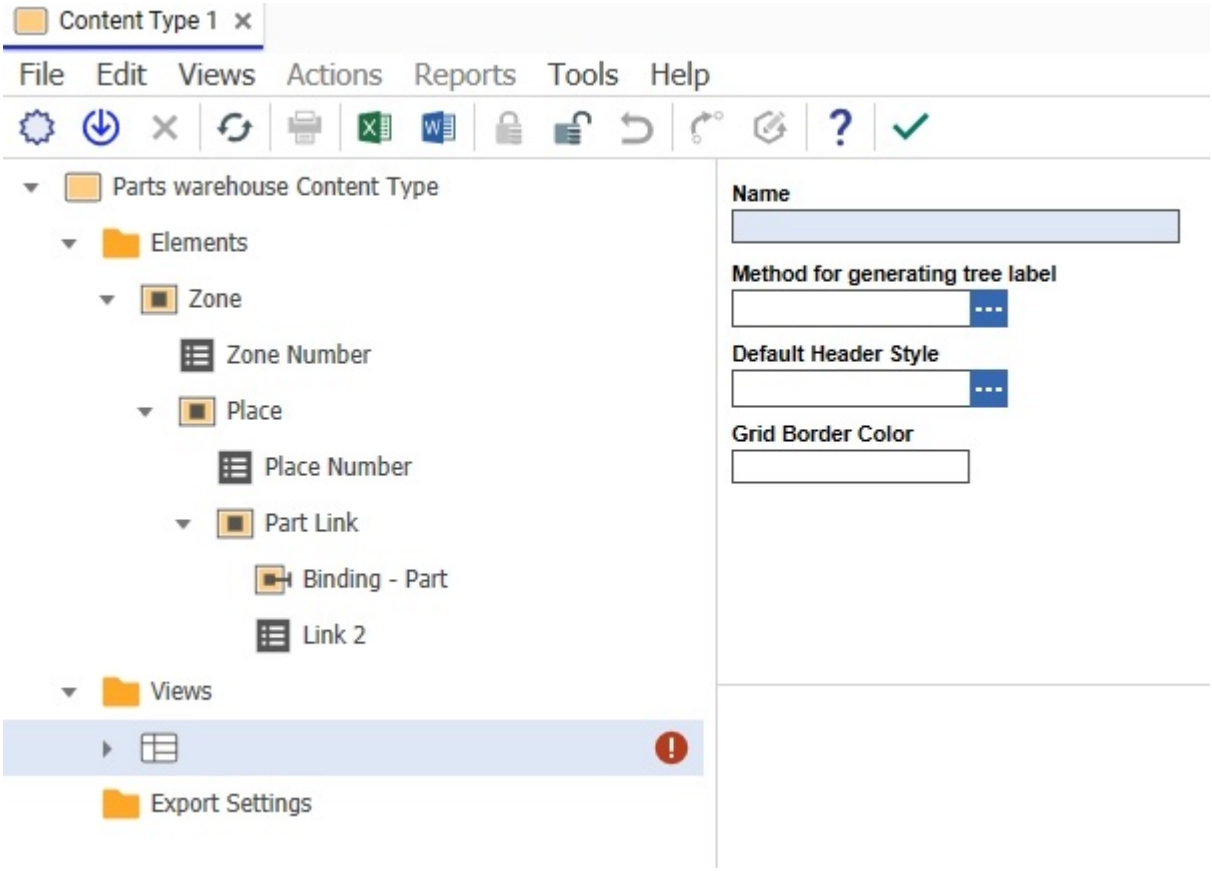
Adding a View for a Content Type

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

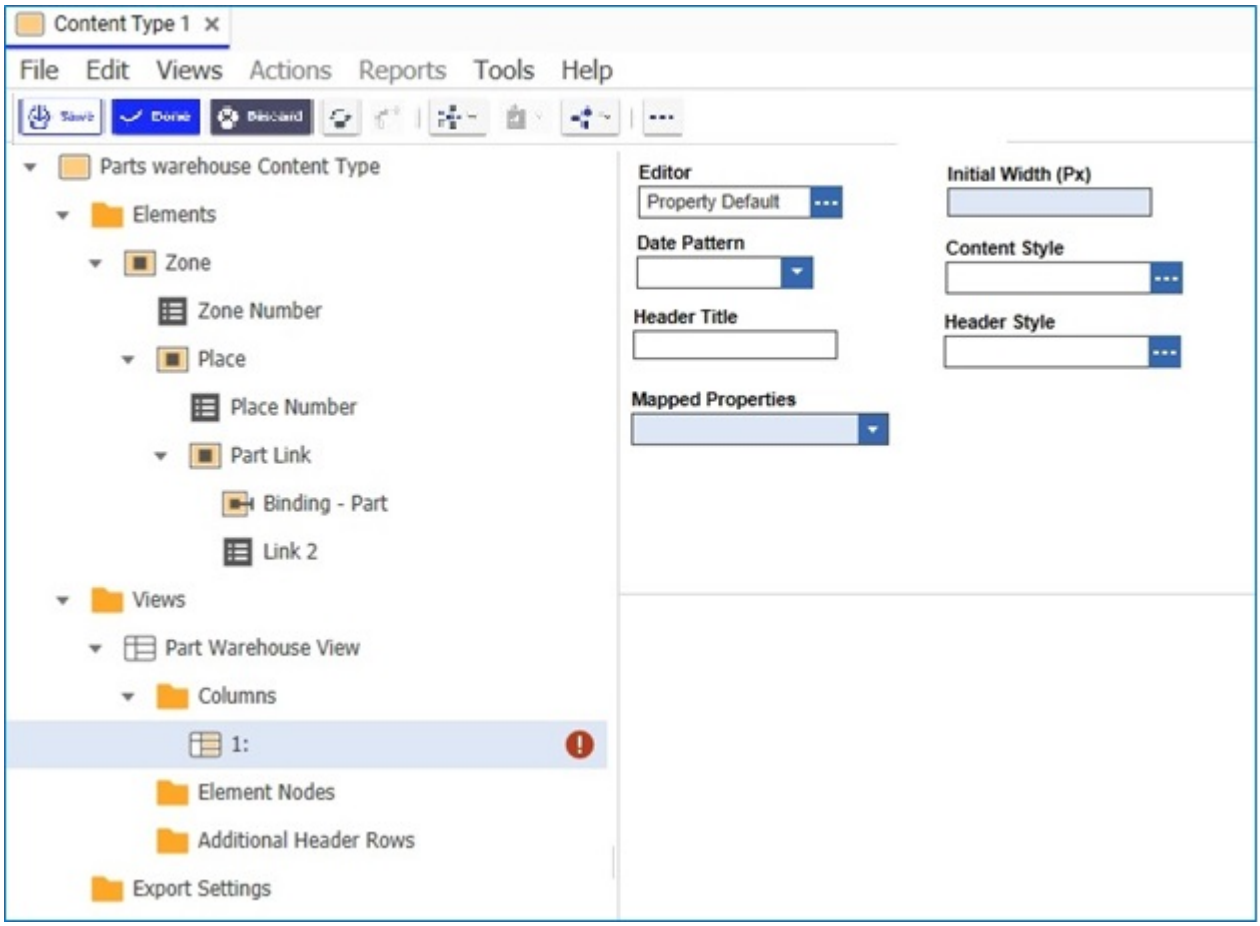
1. Right click the **View** folder and select **Add View** from the context menu. The Select ItemType dialog appears:



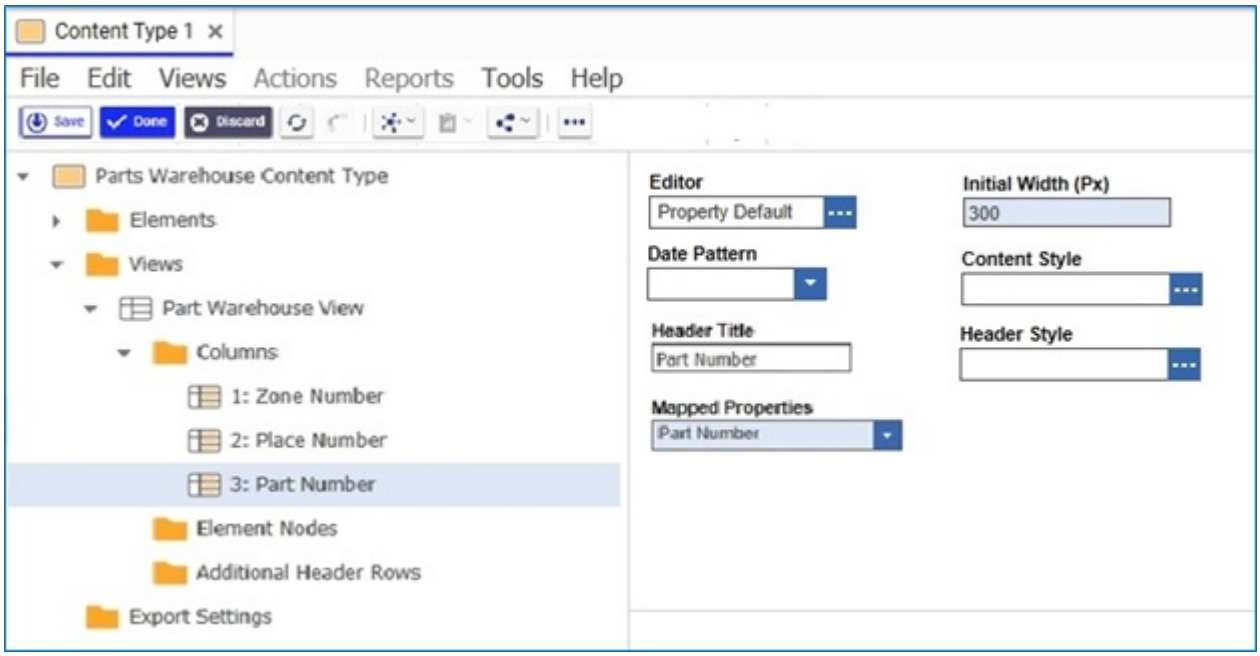
2. Click **OK**. The Tabular View page appears.



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.



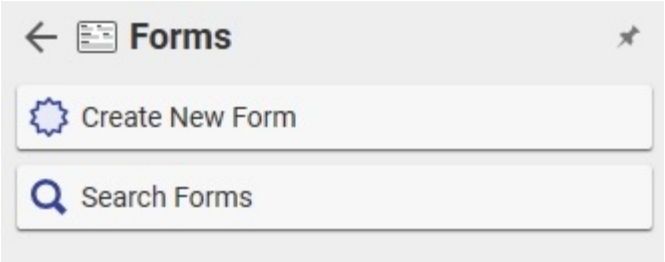
6. Enter **Zone Number** in the Header Title field.
7. Select **Zone Number** from the dropdown list in the Mapped Properties field.
8. Enter **300** in the Initial Width field and click **Save**.
9. Right click 1: Zone Number and select **Add Column** from the context menu to add another tabular view.
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**.



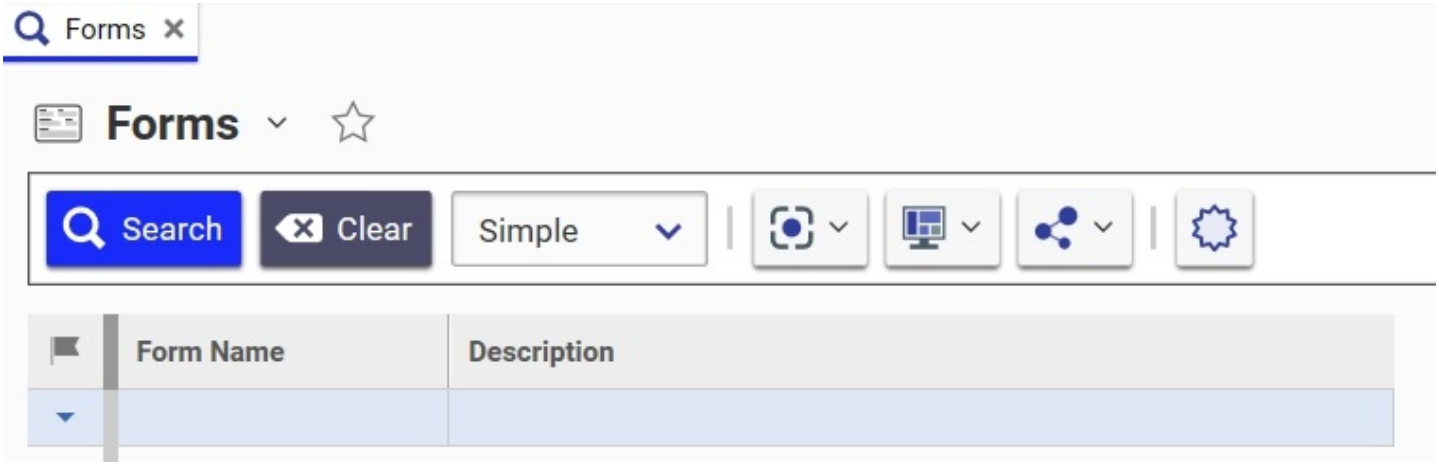
15. Save, Unclaim and close **Parts Warehouse Content** Type.

Adding an "OnFormPopulate" Event and Name Property

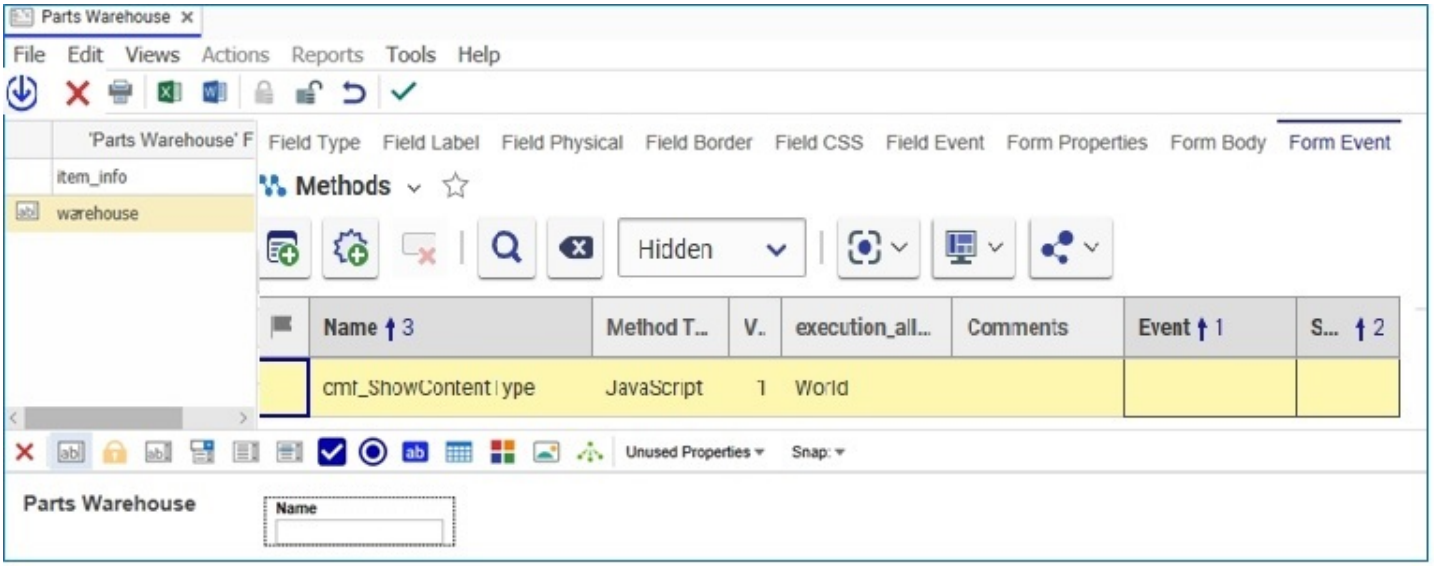
1. Select **Administration-> Forms**. The following menu appears:



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



3. Open the Parts Warehouse form and click null to claim the form for editing.
4. Select the Form Event tab and click the Add Methods icon null . 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).



8. Save, Unclaim and Close the Form.

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 null 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.

Configuration is completed.