Aras Innovator 2025 Release

Extended Classification

Document #: *D-008107*

Last Modified: *5/27/2025*

Copyright Information

Copyright © 2025 Aras Corporation. All Rights Reserved.

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

E-mail: support@aras.com

Website: https://www.aras.com/

Notice of Rights

Copyright © 2025 by Aras Corporation and/or its affiliates. All rights reserved.

This document is protected by U.S. and international copyright laws and conventions. No copyright may be obscured or removed from this document. This document may not be modified or altered, or reproduced or transmitted in any form, without the explicit permission of the copyright holder.

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

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

Notice of Liability

THIS DOCUMENT IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY, AND THE CONTENTS HEREOF ARE SUBJECT TO CHANGE WITHOUT NOTICE. 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 **5**

1 Overview **6**

2 Creating Extended Properties Using the UI **7**

2.1 xProperty Definition 7
2.2 Explicitly Defined xProperties 9

3 Creating Extended Classification Trees **10**

3.1 xClassification Trees 10

3.1.1 Overriding Properties of an xProperty *11*
3.1.2 Making an xProperty Inactive *12*

4 Extended Permissions **14**

4.1 xProperty Value Permissions 15
4.2 Item Classification Permissions 16
4.3 Explicit Permissions 17

5 Extended UI Configuration **18**

5.1 Displaying xClasses and xProperties on an Item Form 18

6 Search Properties, xClasses, and xProperties **20**

6.1.1 Using the Column Selector in Main and Relationship grids *20*
6.1.2 Searching for xProperties across Multiple ItemTypes *27*
6.1.3 Performing an Advanced Search *29*

7 xProperties in AML **32**

7.1 Using the @set attribute 32
7.2 Explicitly Defining an xProperty on an Item 32
7.3 Resolving Ambiguous Property Names 33
7.4 Assigning an xPropertyDefinition to an ItemType 33
7.5 Using @defined_as to Filter xProperties 35
7.6 Getting Additional xProperty Information 36
7.7 Changing Private Permissions 37
7.8 Querying xProperties across Item Types 38

7.8.1 Filtering Items by Item Type Name *38*

7.9 xProperty of Data Type Item 39

7.9.1 Using $value, @keyed_name, @type in a select attribute *39*

8 Working with the Classification Data Model in AML **41**

8.1 Working with classification data in the context of ItemType 41

9 11.0 SP12 Extended AML Enhancements **43**

9.1 Using the “Is Defined/Is Not Defined” Flags 43

©2025 Aras Corporation All Copyrights Reserved.

9.2 Requesting xProperty Information 44
9.3 Filtering Items by Item Type Name 45
9.4 Filtering Items Using the Condition= In and By Attributes 45
9.4.1 Backward Compatibility for Item Data Types 45
9.4.2 Adding xProperties to any PolyItem Type 46
9.4.3 Filtering Items by xClass and Descendants 46
9.4.4 Using [<filter_expression>] 46
10 xClass Search API 47
10.1 Extending the SearchMode base class 47
10.2 Enabling xClass Search for Custom Search Mode 47
10.3 Extending Custom SearchMode to work with xClass Search 47

©2025 Aras Corporation All Copyrights Reserved.

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/

©2025 Aras Corporation All Copyrights Reserved.

1 Overview

Aras Innovator's Extended Classification feature enables non-administrative users to classify ItemTypes without modifying them. Users are able to assign properties to Items based on the item's extended classification. They can also perform class-based searches either programmatically or through the UI.

Dynamic properties are referred to as Extended Properties (xProperty ItemType). Dynamic classes are referred to as Extended Classes (xClass Item Type). You can assign xProperties to multiple xClasses. xProperties assigned to Parent xClasses are inherited by the child xClasses. You can also assign xProperties directly to multiple ItemTypes as explicit xProperties.

Users with the correct permissions can create and maintain Extended Property Definitions and Classification Trees either programmatically or through the UI. End users can assign multiple xProperties to one item. They can also:

- Classify items
- Set values for the xProperties associated with a specific item
- Use xProperty and xClass values to search for items

©2025 Aras Corporation All Copyrights Reserved.

2 Creating Extended Properties Using the UI

Extended Properties (xProperties) are property definitions that are isolated from any one ItemType definition. Members of either the “Administrators” or the “Classification Administrators” group identities can create these. Members of these groups can also specify unique permissions for each xProperty.

2.1 xProperty Definition

To define xProperties, select **Extended Classification → xProperties** in the TOC.

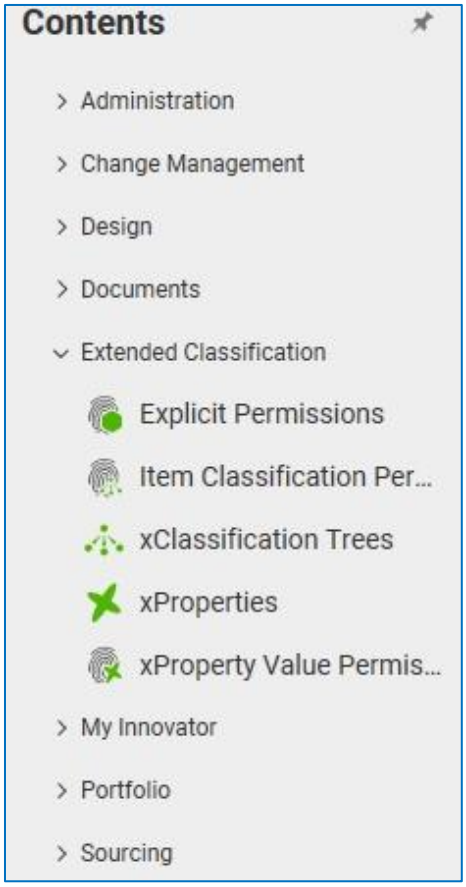

Figure 1.

The menu shown in Figure 2 appears.

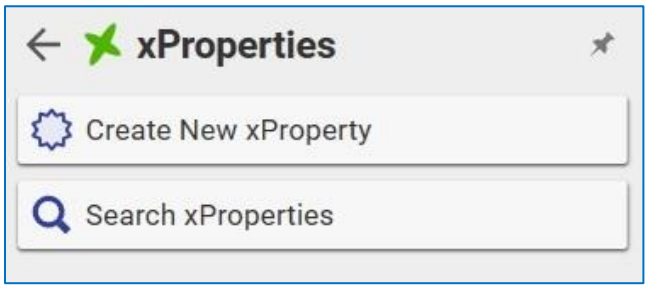

Figure 2.

©2025 Aras Corporation All Copyrights Reserved.

Click **Create New xProperty**. A screen similar to the following appears. You define an xProperty in the same way as an ItemType property. Definitions include information such as the name, label, and data type.

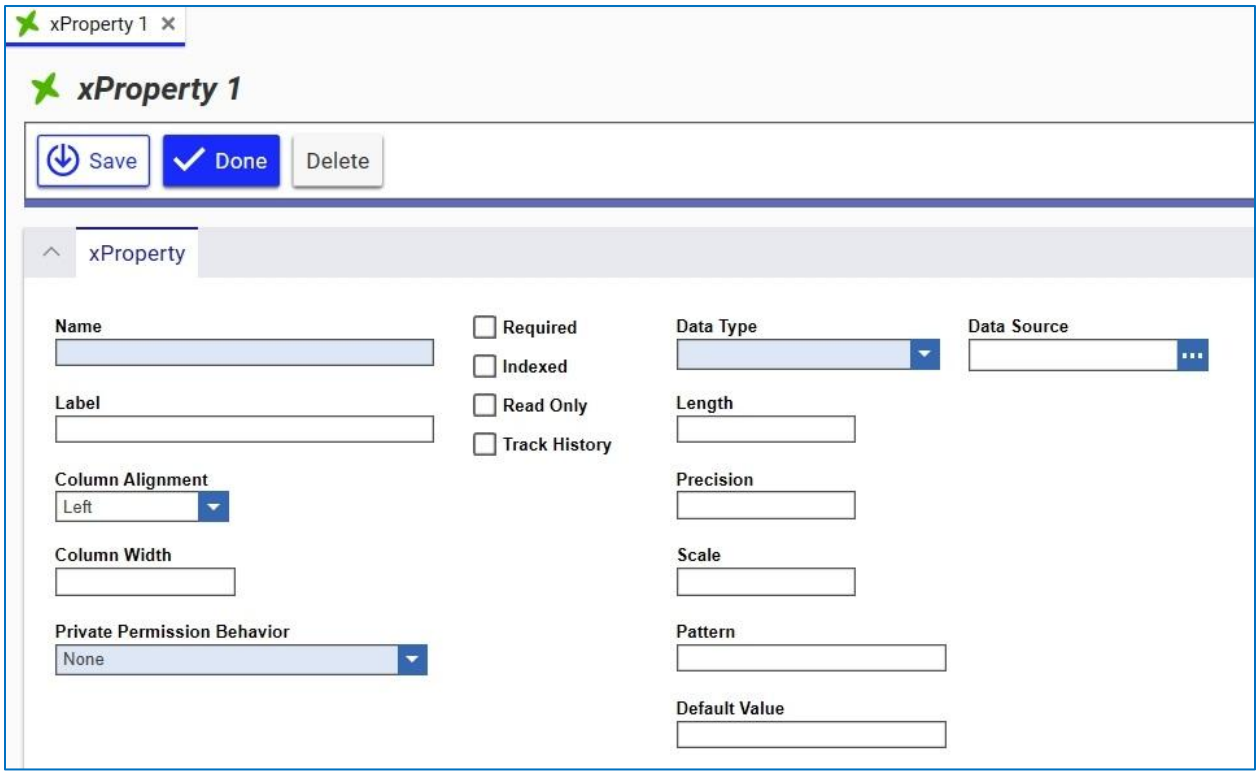

Figure 3.

Note: Only the following data types are currently supported:

Boolean
Color
Color List
Date
Decimal
Float
Integer
Item (Fixed Behavior Only)
List
Multi Value List
String
Text

Private Permission Behavior is discussed in section 4.1

©2025 Aras Corporation All Copyrights Reserved.

2.2 Explicitly Defined xProperties

You can add xProperties directly to an ItemType, separate from any xClassification definition. To explicitly link an xProperty to an ItemType, open the ItemType's definition and add the xProperty under the xProperties tab.

You can then either tie the permissions of this xProperty to the standard ItemType permissions or to its own unique permissions. There are two settings for unique permissions, xProperty **Value Permissions** and **Explicit Permissions**. If you set the Permission Behaviors to **No Check**, the standard Item permissions apply. Setting the behaviors to **Check** enables the configured permissions. Permissions are described in more detail in section 4.

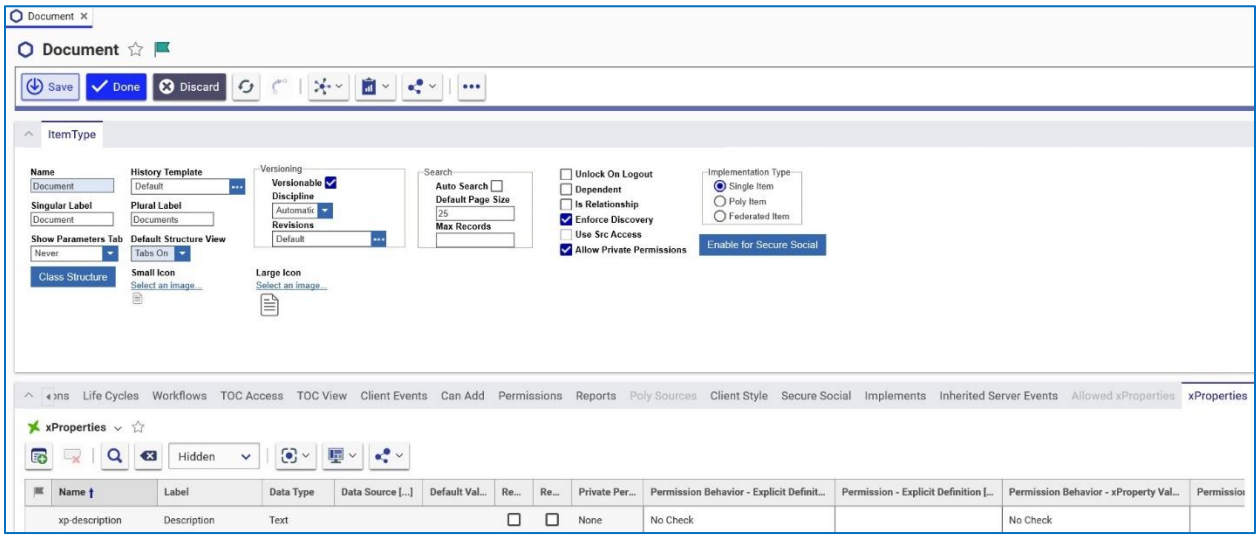

Figure 4.

Note:	When changing the Permission Behavior, you may need to rebuild the xProperty secured function using the following procedure: 1. Edit the xProperty definition and toggle the "Private Permission Behavior" setting. 2. Save the xProperty. 3. Toggle the behavior back to its original setting and save the xProperty once more.

©2025 Aras Corporation All Copyrights Reserved.

3 Creating Extended Classification Trees

xProperties are primarily linked to ItemTypes through their Extended Classifications (xClassification/xClass). These are defined in the TOC under **Extended Classification** --> **xClassification Trees**.

The xClassification Tree is a hierarchical structure containing sub-classes.

3.1 xClassification Trees

xClassification Trees are defined in two pieces. The standard form defines basics such as the name and identifying number of the tree as well as two additional configurations:

- Restrict Selection To Only Leaf Classes

  When selected, you can only select leaf classes in the xClassification Tree. These are classes that have no child classes.

- Restrict Selection To A Single Class

  When selected, you can only choose one Extended Classification for each Item. Disabling this flag allows you to assign multiple xClasses to a single Item so that all of those xClasses properties are available.

The relationship tab “Item Types” defines the ItemTypes where this extended classification tree is available.

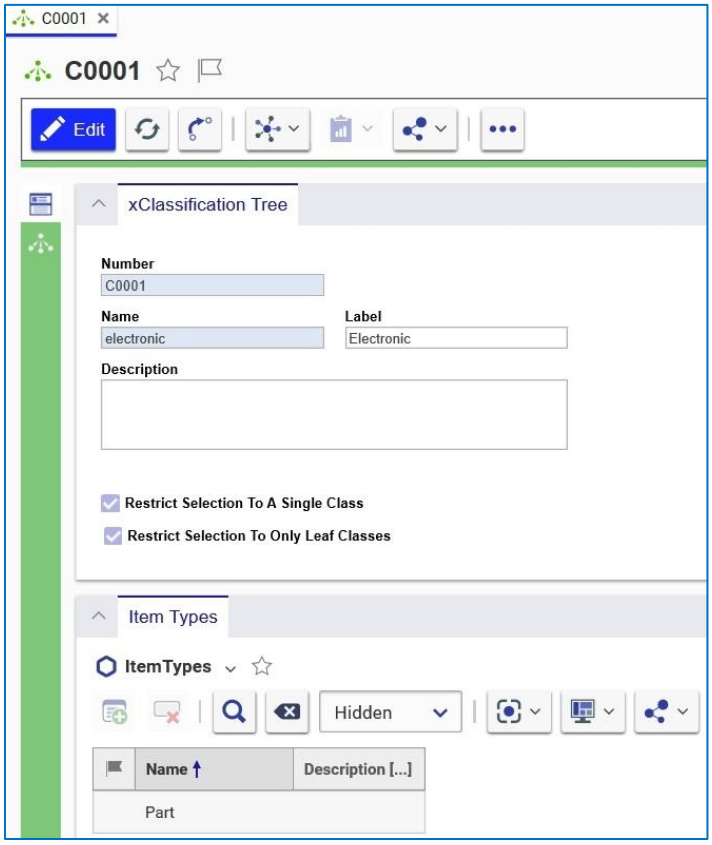

Figure 5.

©2025 Aras Corporation All Copyrights Reserved.

To define the xClassification Tree, select the **Show Editor** icon in the green sidebar. This displays an Editor that enables you to perform the following functions:

- Create the Tree structure- To add a new xClass, right click on the top level of the tree and select **Insert Sub-class**. The top level xClass shares the name of the xClassification Tree.
- Assign xProperties to specific xClasses – Select the xClass from the tree in the left column and select

  Pick Related from the dropdown list in the toolbar on the right. Click the **New Relationship** icon . The xProperty search dialog opens, enabling you to select the desired xProperty. An xClass can have multiple xProperties assigned to it.

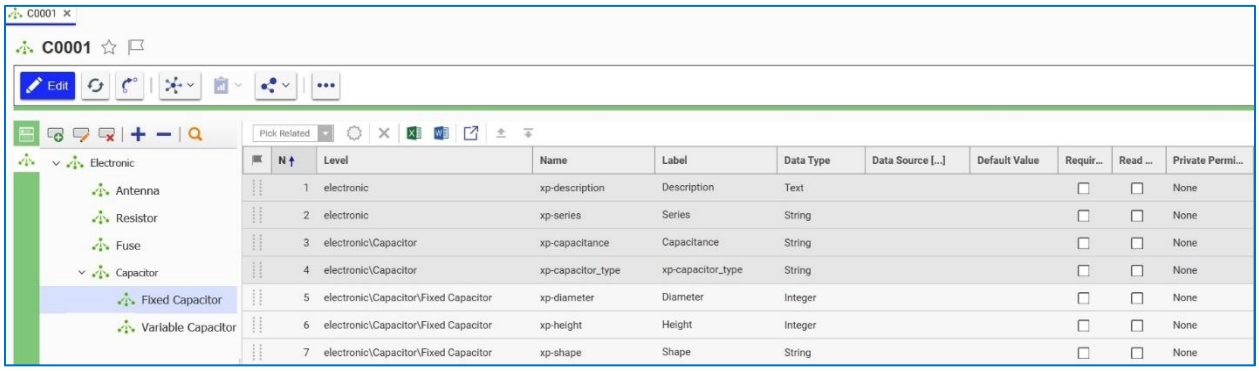

Figure 6.

In this xClassification Tree seven xProperties are assigned to various xClasses. When you select a node, all included xProperties appear in the right grid. Those with a grey background are inherited from higher level xClasses. The xProperties defined for the selected xClass appear with a white background.

- Within the grid, it is possible to drag and drop rows by clicking the in a particular row to reorder the xProperties. The order shown will be the same as the order that appears on an Item's form when you select the xClass to classify the item. You can also use the Move Up and Move Down buttons in the Editor Toolbar to reorder rows.

3.1.1 Overriding Properties of an xProperty

For a given xProperty the properties Label, Default Value, Required, and Read Only are created on a global scope. The Classification Administrator can override the value of these four properties when the xProperty is assigned to an xClass. The new values (Overridden) are the class specific properties of the xProperty.

Overriding the properties of an xProperty can be done in the xClass Tree Editor.

©2025 Aras Corporation All Copyrights Reserved.

When you select a row in the grid, toggle buttons appear in the Label, Default Value, Required, or Read Only Solutions columns for that xProperty, as shown in the following screenshot:

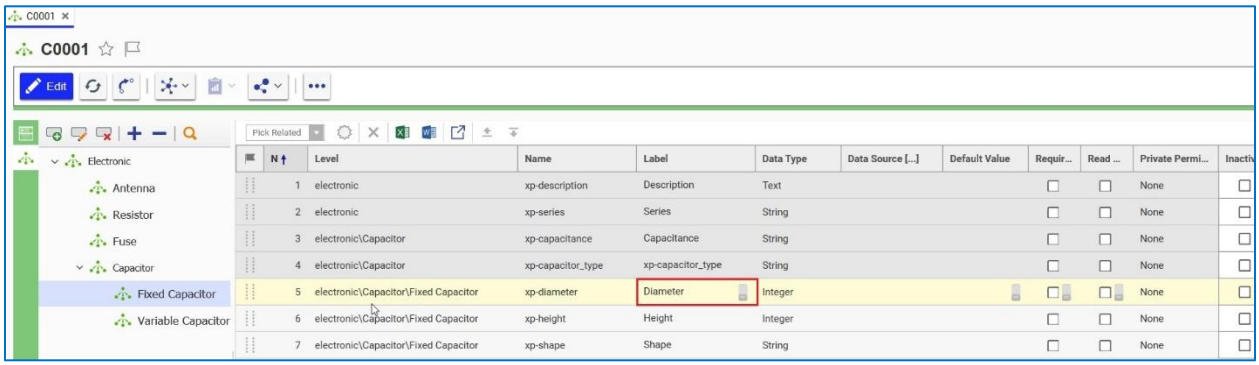

Figure 7.

The values that appear in these cells are the Global values. When you click a toggle button, the value that appears in the cell is replaced with a blank space as shown here:

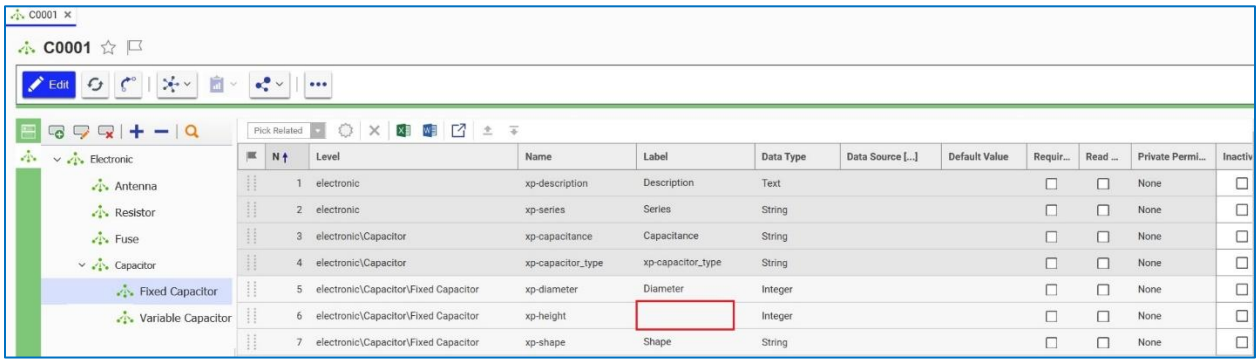

Figure 8.

Click the cell to enter a new value. This new value is the Override value. Click on the toggle button again to return to the Global value.

xProperties that are assigned to a specific xClass appear in the grid with a white background. Inherited xProperties appear with a grey background.

It is possible to override property values for xProperties that are inherited from a higher xClass. All child classes with inherited xProperties will inherit the values of their immediate parent class – regardless of whether that value was overridden or inherited from a higher class.

3.1.2 Making an xProperty Inactive

You can make an xProperty associated with an xClass inactive by selecting the Inactive flag for that property as shown here:

©2025 Aras Corporation All Copyrights Reserved.

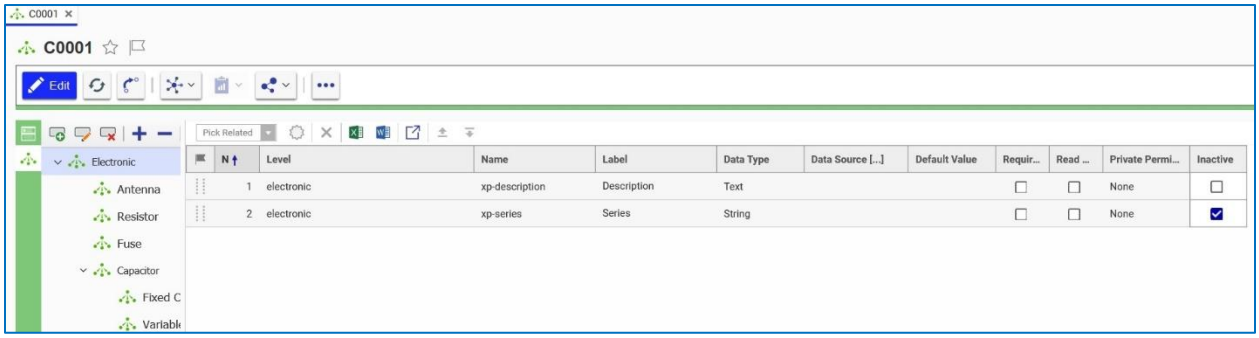

Figure 9.

When you make an xProperty inactive, end users will not be able to see it in the UI. This is true for both existing items that have the xProperty assigned to them and new items that you have created and classified. The xProperty and its associated value are not deleted and you can re-activate them by deselecting the Inactive checkbox.

When you make an xProperty inactive, it no longer appears in the column selector, the main grid, or the Relationship grid. It also will not appear in the Advanced Search Property dialog box. If you stored the xProperty to a saved search, you will need to update it so it does not appear when you select that search for an item.

©2025 Aras Corporation All Copyrights Reserved.

4 Extended Permissions

To be able to use an xClassification Tree and xProperties, you must set Extended Permissions on the xClassification Tree Form for each ItemType. In the following case, two permissions are set for the Part ItemType by right-clicking **Part → Pick xProperty Value Permission** and **Pick Item Classification Permission**.

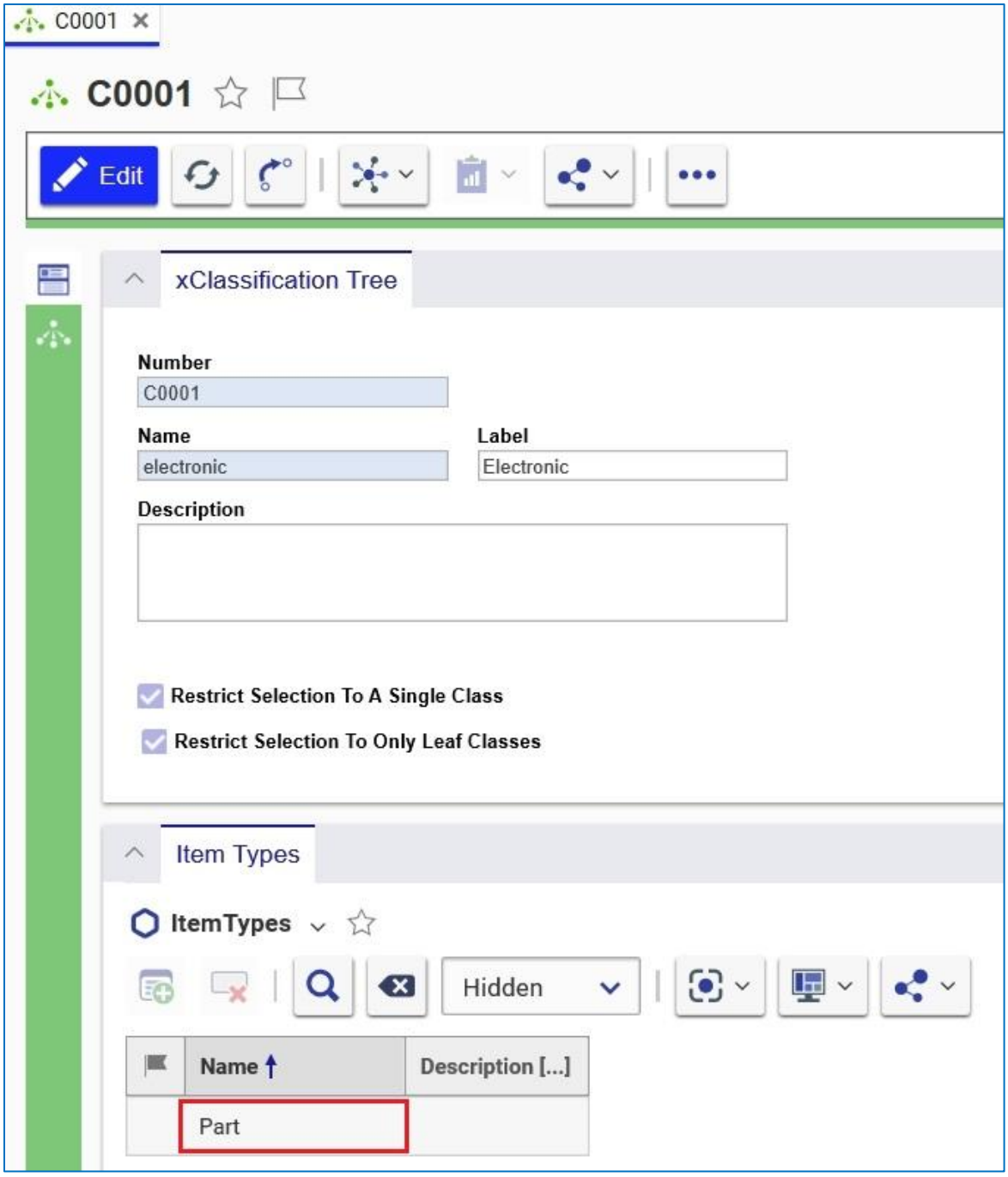

Figure 10.

©2025 Aras Corporation All Copyrights Reserved.

Set the following permissions for explicit xProperties: **Explicit Permission** and **xProperty Value Permission**. These permissions should be set for each xProperty in the xProperties tab of the ItemType.

4.1 xProperty Value Permissions

Specific xProperty permissions are defined under **Extended Classification → xProperty Value Permissions** in the TOC. These define Get, Update, and Can Change access to the selected Identities:

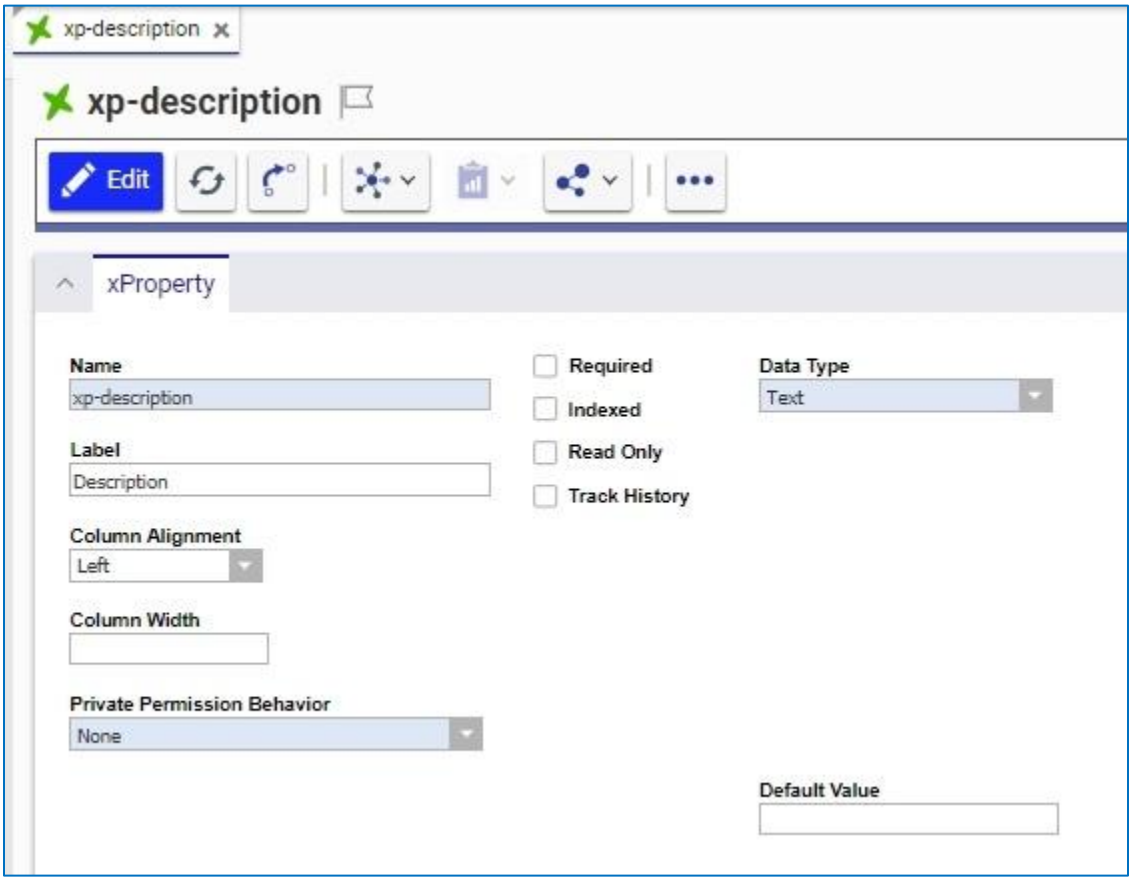

Figure 11.

To define the permission for the xClassification Tree, right click on the ItemType under the **Item Types** tab on the form view and select either **Create xProperty Value Permission** or **Pick xProperty Value Permission**. Once selected, you have the option to view or replace the selected permission.

If the Private Permission Behavior on the xProperty form is set to “Any”, then you can set an xProperty Value Permission on a specific Item’s xProperty. This can be done using the following AML:

```
<AML>
   <Item type="Part" id="816AEDE3506042C38211796A3581F4B9" action="edit">
      <xp-shape set="permission_id"
      permission_id="A518B142B9CB4013A85392AA60AB7899"/>
   </Item>
</AML>
```

©2025 Aras Corporation All Copyrights Reserved.

4.2 Item Classification Permissions

Select **Extended Classification → Item Classification Permissions** in the TOC to define specific xClass permissions. These define whether users can see the current xClass as well as whether they can set or clear the Extended Classification on an Item.

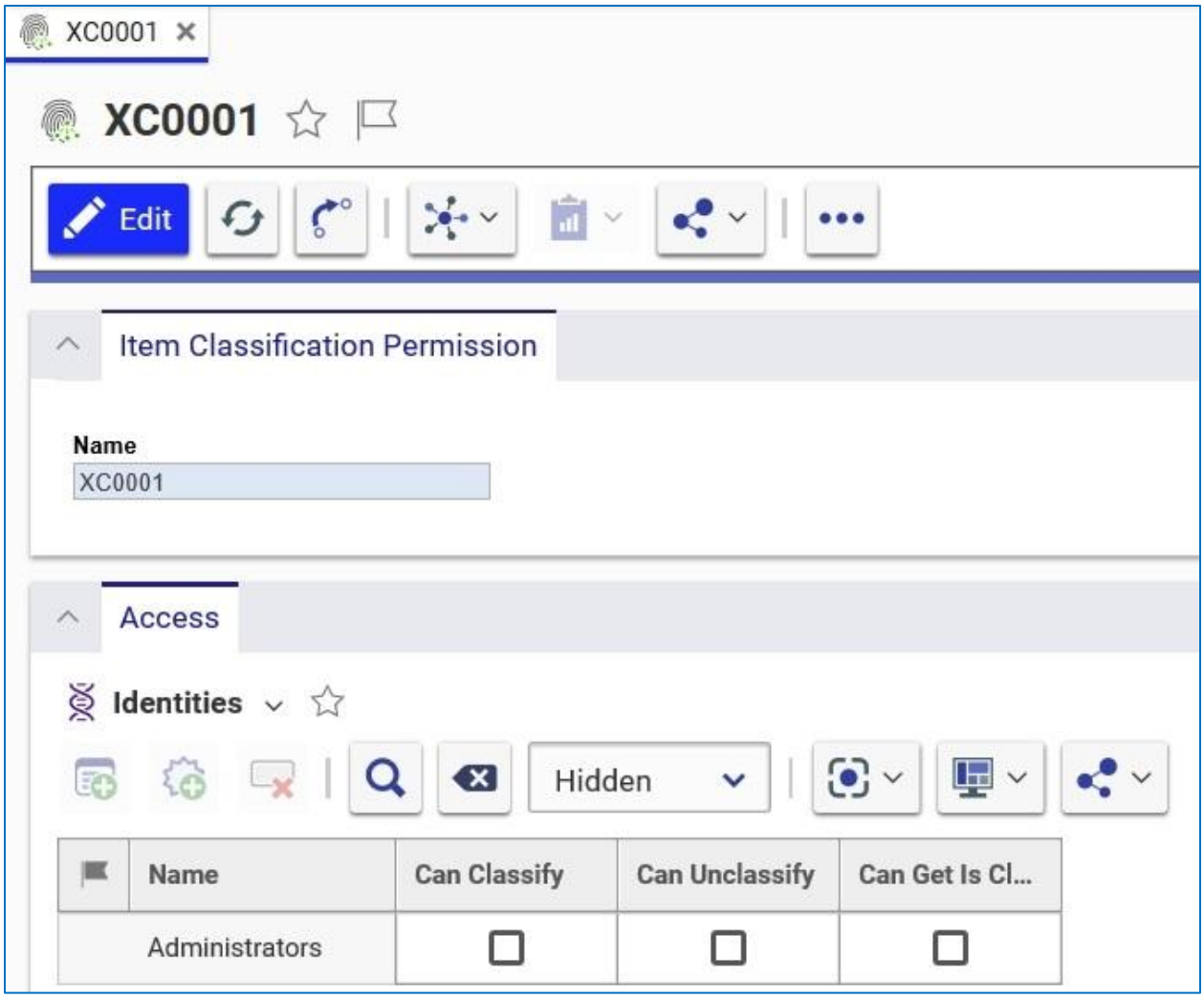

Figure 12.

To define the xClass permissions, right click on the ItemType on the **Item Types** tab in the form view and select either **Create Item Classification Permission** or **Pick Item Classification Permission**. Once selected, you can either view or replace the selected permission.

©2025 Aras Corporation All Copyrights Reserved.

4.3 Explicit Permissions

Explicit xProperty permissions are defined under **Extended Classification** → **Explicit Permissions** in the TOC. These specify an Identity's access to explicitly define or undefine an xProperty as well as the access to query xProperties that are explicitly defined.

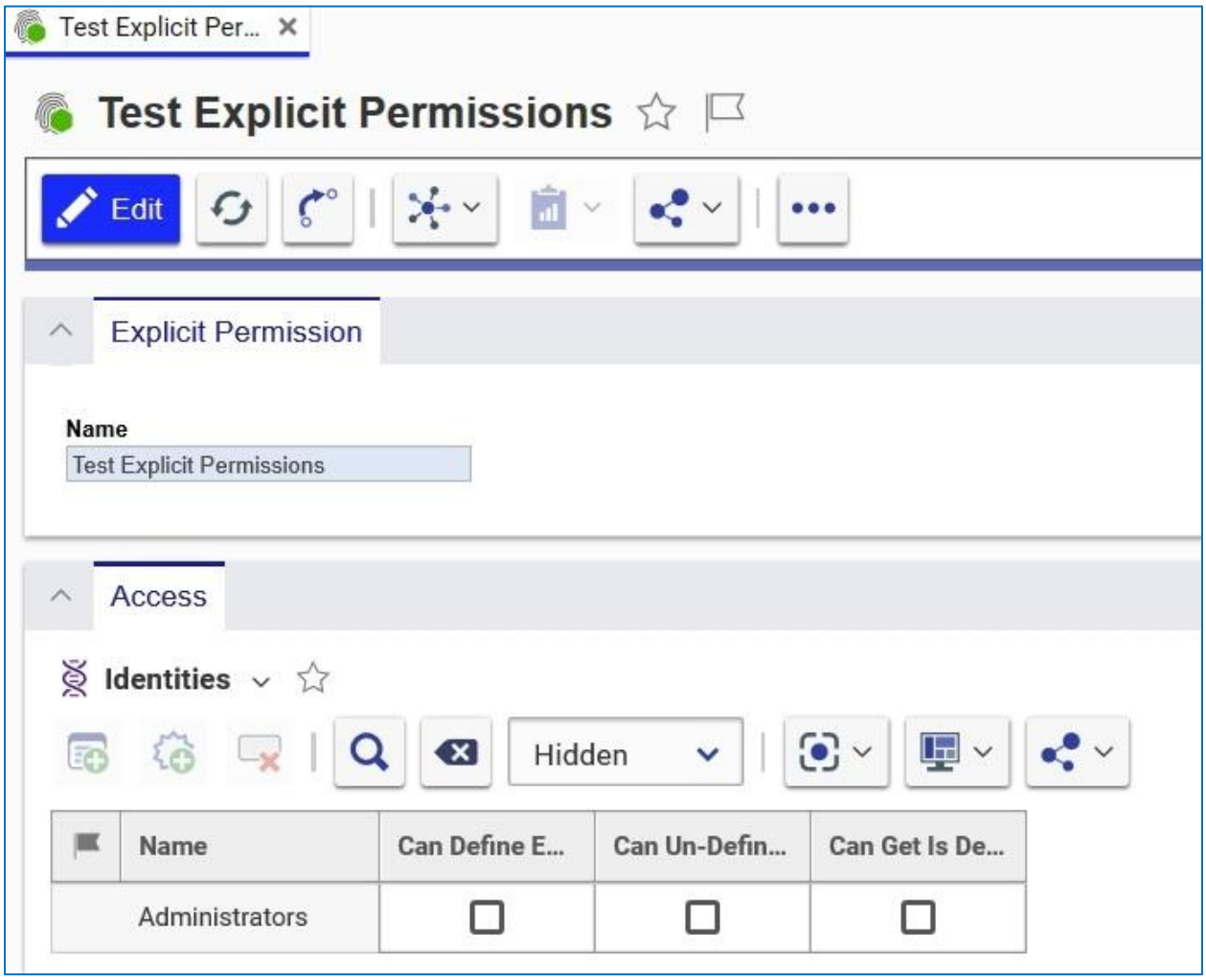

Figure 13.

©2025 Aras Corporation All Copyrights Reserved.

5 Extended UI Configuration

5.1 Displaying xClasses and xProperties on an Item Form

To display xClasses and xProperties on the Item's form, you must first edit the form and insert a new xClass control by clicking the ellipses in the xClass field.

Note: You must have Administrator rights to do this.

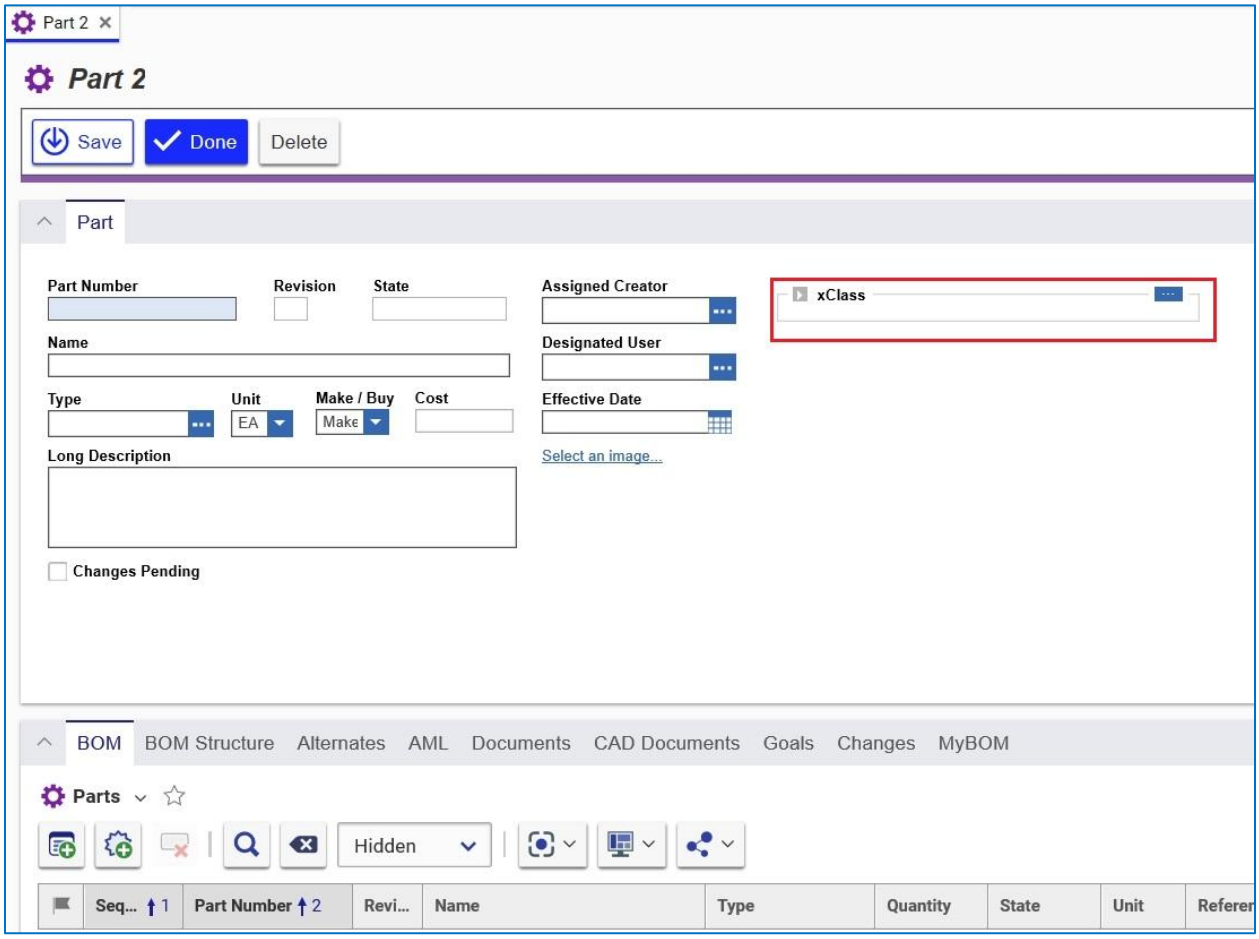

Figure 14.

©2025 Aras Corporation All Copyrights Reserved.

Once you add the field you can pick the xClass using the ellipsis icon on the top right of the xClass field. The Classification dialog box appears:

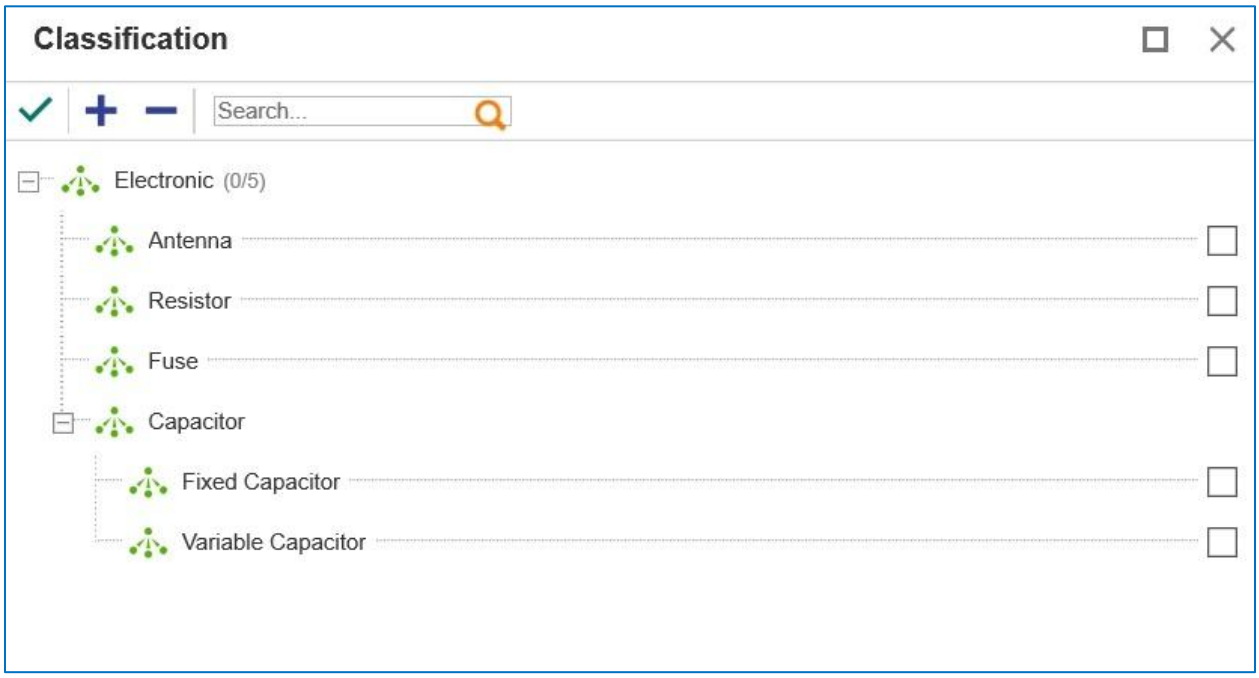

Figure 15.

Properties for the selected xClasses appear in the field under a dropdown for each xClass. You can expand xClasses separately or you can expand them all at once by selecting the blue arrow sign at the top left of the field:

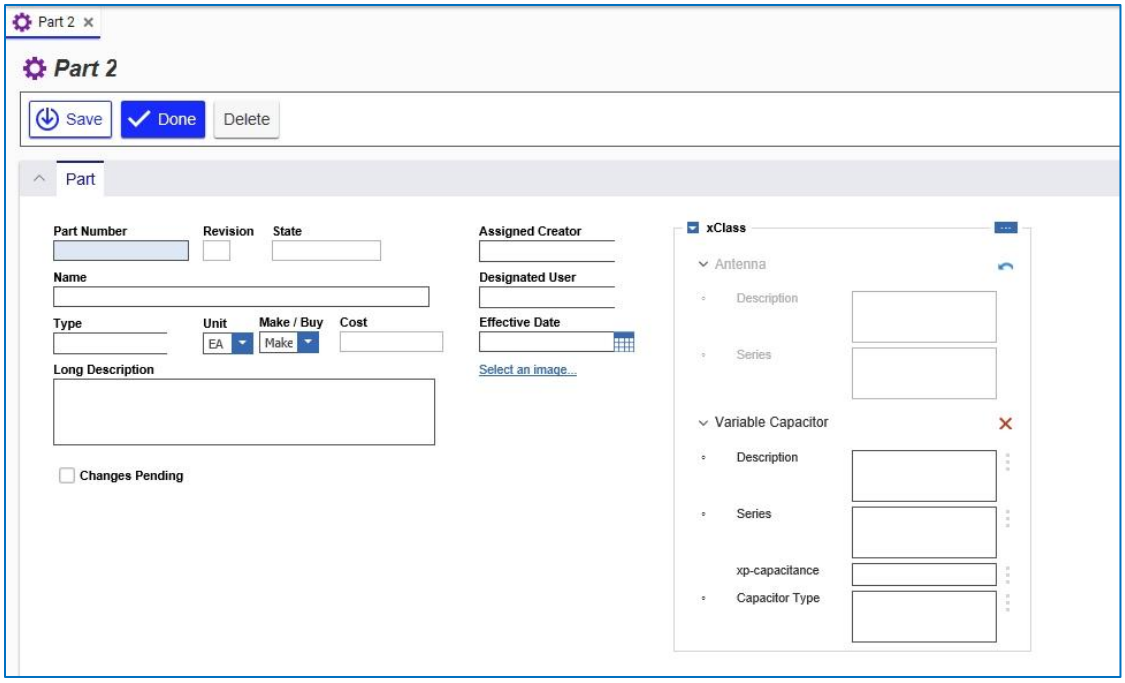

Figure 16.

©2025 Aras Corporation All Copyrights Reserved.

6 Search Properties, xClasses, and xProperties

You can search for properties, xClasses, and xProperties associated with items using Simple, Advanced and AML search. The xClass and xProperty criteria can also be saved to a Saved Search.

6.1.1 Using the Column Selector in Main and Relationship grids

The Main and Relationship grid toolbars contain the Refine Search icon . When you click the icon, a dialog box similar to the following appears. The dialog contains 2 columns – Properties and Classifications.

Note: The data displayed in the following dialog uses example data.

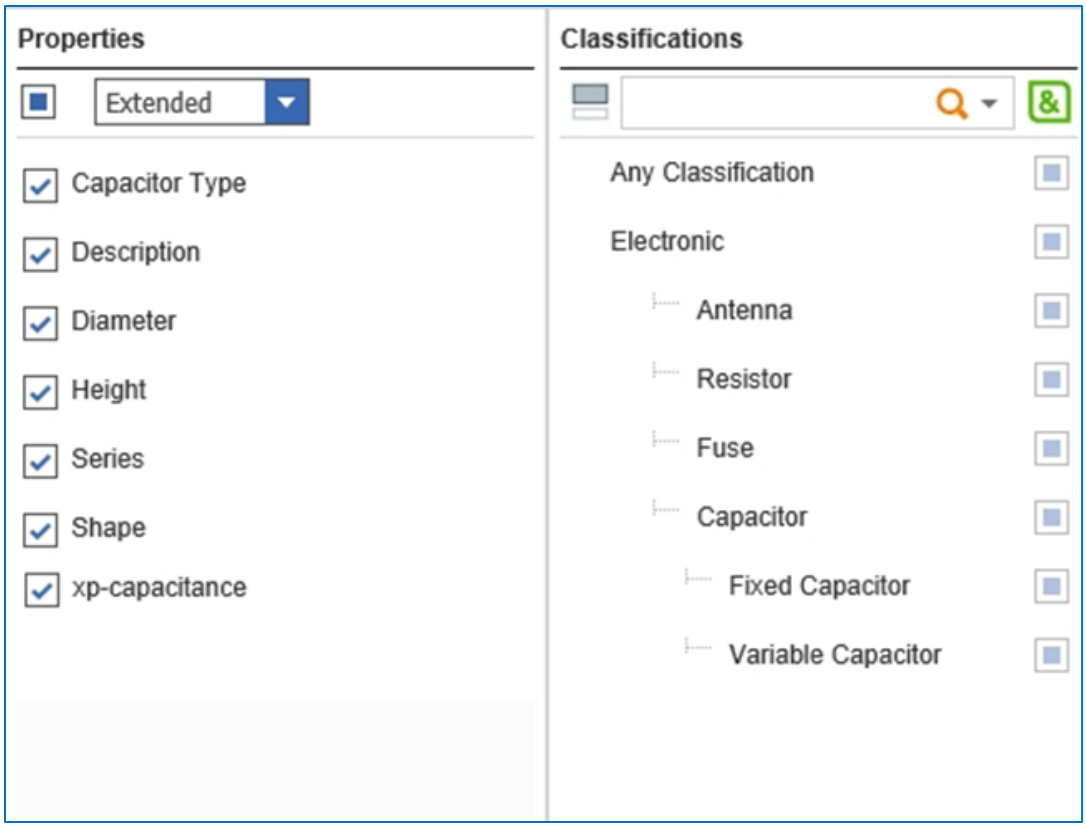

Figure 17.

In the Main Grid, the Properties column displays a list of properties associated with the item. Select one of the following from the dropdown list:

- **All** (the default) displays the standard properties, explicitly-defined xProperties, and extended properties associated with an item.
- **Standard** displays the standard properties and explicitly defined xProperties.
- **Extended** displays all of the xProperties assigned through the xClassification trees.

In the Relationship grid the Properties column displays a list of properties associated with the item. Select one of the following from the dropdown list:

©2025 Aras Corporation All Copyrights Reserved.

- **All** (the default) displays all the properties.
- **Standard** displays all Properties where hidden2 = false **and** explicitly defined xProperties on the **Related ItemType**
- **Relationship** displays Properties where hidden2 = false as well as explicitly defined xProperties **and** xProperties assigned through xClass Trees for the **Relationship ItemType**
- **Extended** displays all the xProperties that are assigned through xClassification Trees for the **Related ItemType**

The standard properties that you select appear as column headings in the Main and Relationship grids. Deselecting properties removes them from the grid.

The Classifications column displays the xClassification Tree associated with an item.

To search for several xProperties or xClasses, click the AND icon . Click the OR icon  to search for a particular xProperty or xClass.

- For example if the User searches for xClass X OR xClass Y the search should only return Items classified as X or items classified as Y, or Items classified as both X and Y. And Items classified by X and Z. And Items classified by Y and Z. And Items classified by X, Y and Z. etc. As long as an item is classified by X or Y, it should be returned no matter what else it is classified with.
- If the User searches for xClass X AND xClass Y the search should return Items classified as both X and Y and Items classified by X, Y and Z.

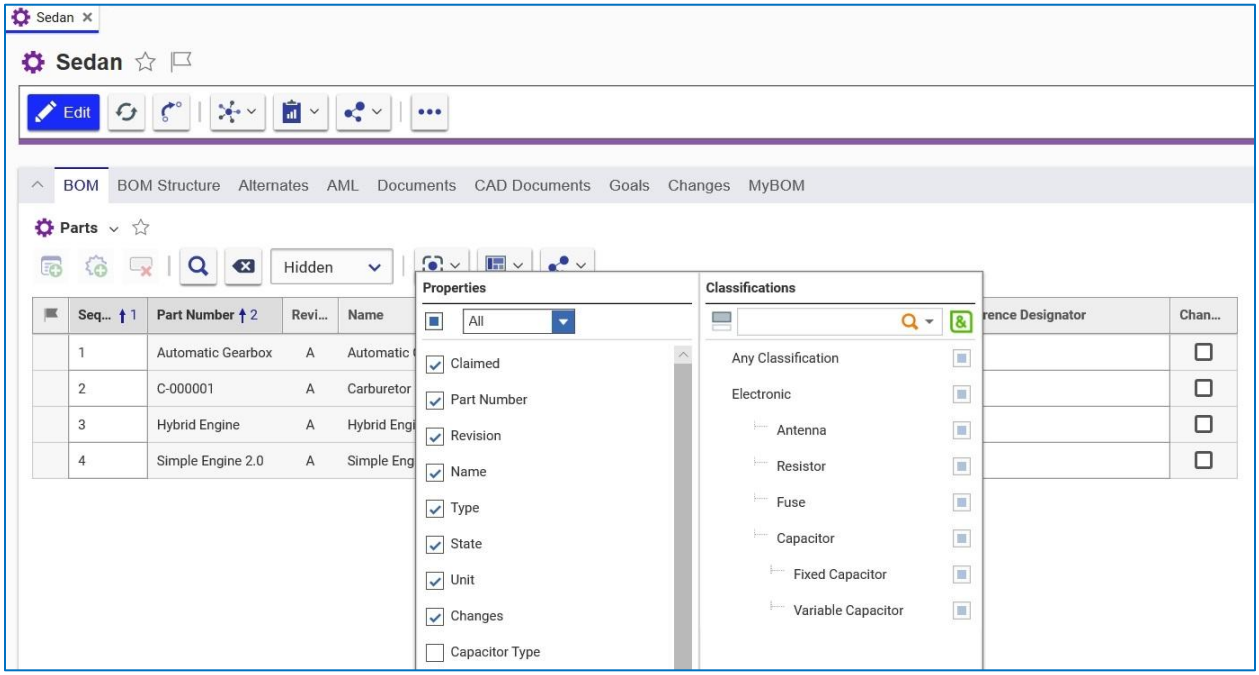

Figure 18.

6.1.1.1 Filtering the Property Column List by xClasses

You can search for xProperties that are only associated with a particular xClass by filtering xProperty Columns by selecting xClasses. When you select an xClass in the Classifications Column, the xProperties associated with that xClass appear in the Properties column. If you select multiple xClasses, the xProperties for all of the xClass selections are displayed in the Properties Column. Unselecting the "xClass Filter" control for xClasses causes the assigned xProperties for those xClasses to not appear in the Properties column.

©2025 Aras Corporation All Copyrights Reserved.

As shown in the following screenshot, selecting the filter for the “Fixed Capacitor” xClass results in the xProperties associated with the Fixed Capacitor xClass appearing in the Properties column. The filter icon for each xClass is located to the right of the xClass name. The default state of the icon is grey (unselected). When you select the filter for a given xClass, the filter icon appears in orange.

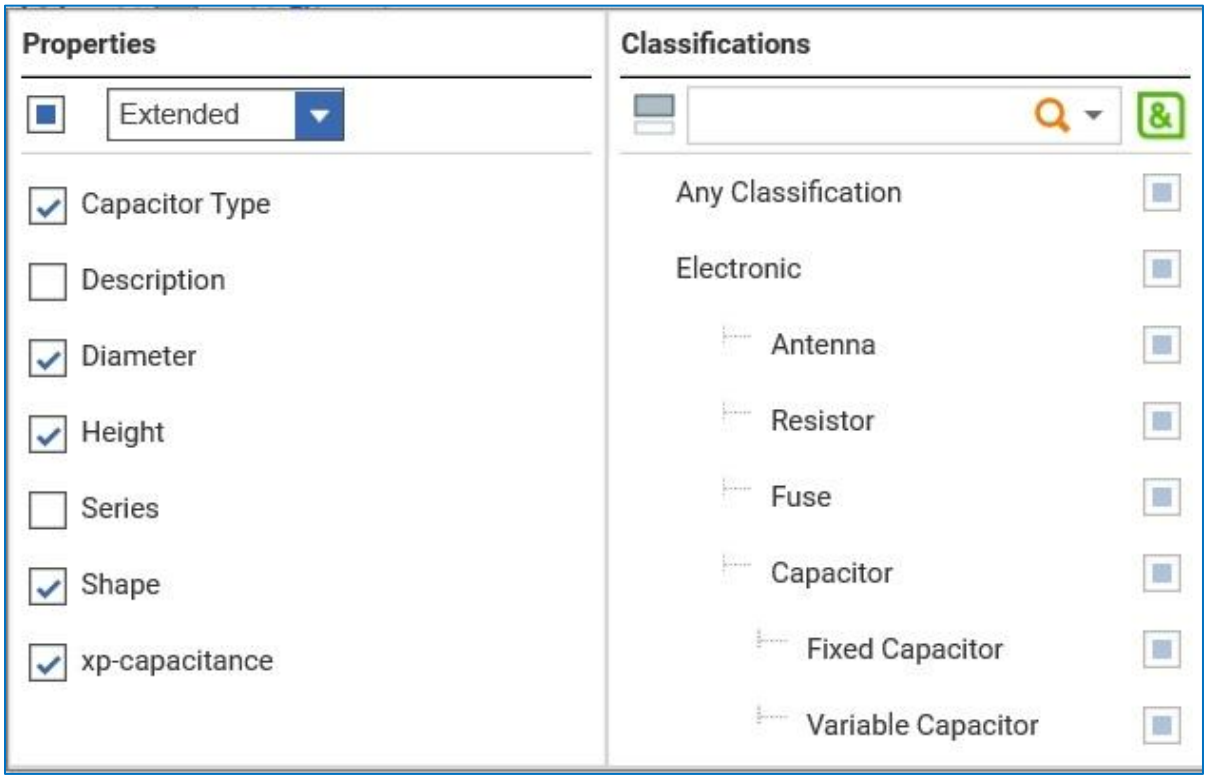

Figure 19.

©2025 Aras Corporation All Copyrights Reserved.

6.1.1.2 xClass Search Options

In the Classifications Column of the Column Selector, the following states (1-7, shown in the diagram below) are available for selecting the xClass to include in the Search criteria.

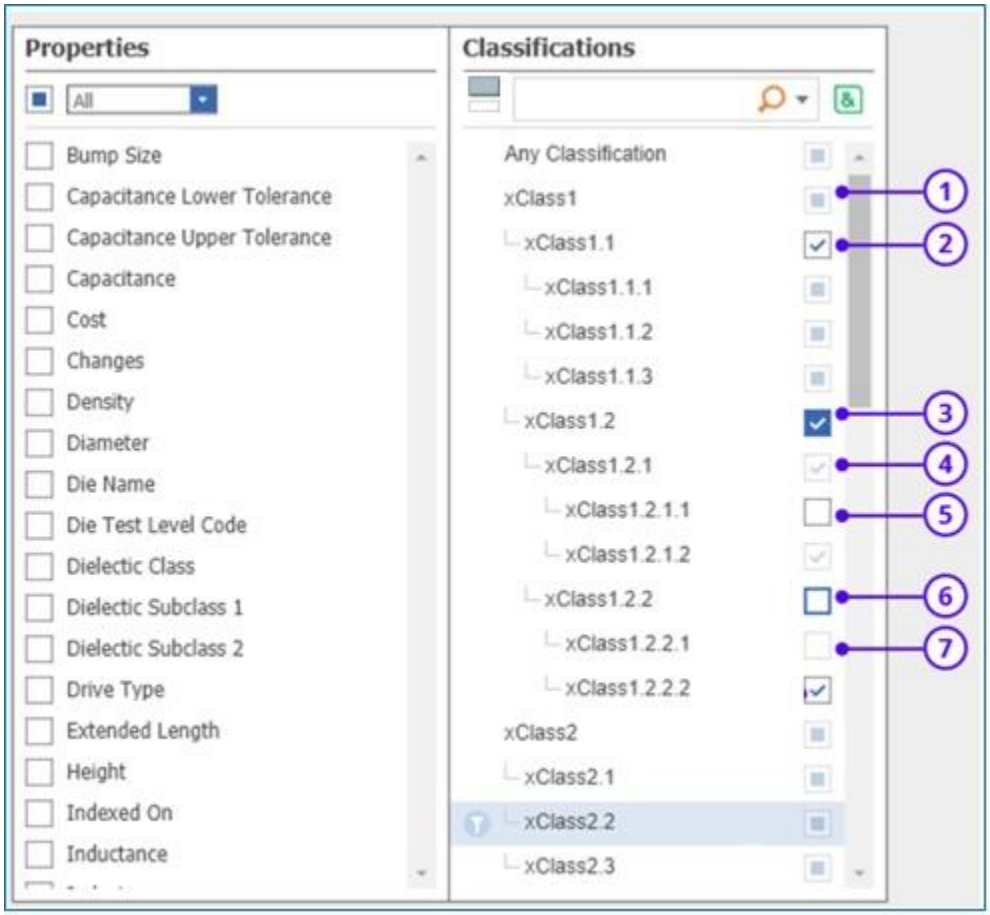

Figure 20.

The check boxes on the right of the xClass name enable you to select the xClass to include in the search criteria. The xClass you select for the search is displayed in the xClass bar. For more information refer to section 6.1.1.3.

Figure 18 shows the different states for the checkboxes. Each of these states is described here:

1. **Unselected + Indeterminate** - This is the default search state for all xClasses. The indeterminate symbol indicates that it is .inactive. Grid results may or may not be classified by this xClass.
2. **Checked** - When you click the box, the check symbol that appears indicates that the xClass has been included as a single search criterion.
3. **Hard Checked** (include subclasses) — When you click an xClass checkbox twice the inverted colors on the check symbol indicate that both this xClass and all associated subclasses are included as search criteria. (Only available while using ‘AND’ logic).
4. **Soft Checked** - The normal check symbol is grayed out to indicate that the xClass is included as a single search criterion. You cannot click to this state. It only appears if a parent xClass was set to “Hard Checked”.
5. **Unchecked** - The empty box indicates that the search results must ‘NOT’ include this xClass.

©2025 Aras Corporation All Copyrights Reserved.

6. **Hard Unchecked** (exclude subclasses) - The blue border around the empty box indicates that the search results must ‘NOT’ include this xClass or any of its subclasses. (Only available while using ‘AND’ logic).
7. **Soft Unchecked** - The empty box indicates that the search results must ‘NOT’ include the xClass. You cannot select this state. It only appears if a parent xClass was set to “Hard Unchecked”.

The Click-through cycle for xClasses that have subclasses where ‘AND’ logic is active starts with the default “Unselected + Indeterminate”. It then progresses through Checked, Hard Checked, Unchecked, Hard Unchecked, then back to Unselected + Indeterminate**.**

The Click-through cycle for all other conditions, beginning with the default Unselected + Indeterminate is Checked, Unchecked, and then back to Unselected + Indeterminate.

Clicking anywhere outside the menu closes the menu.

Clicking **Run Search** from the search toolbar executes the xClass search criteria and returns the results.

Clicking **Clear Search Criteria** from the search toolbar, clears all xClass search criteria.

Any Classification – Users can also search Items with Any or No Classification as shown in the following diagram.

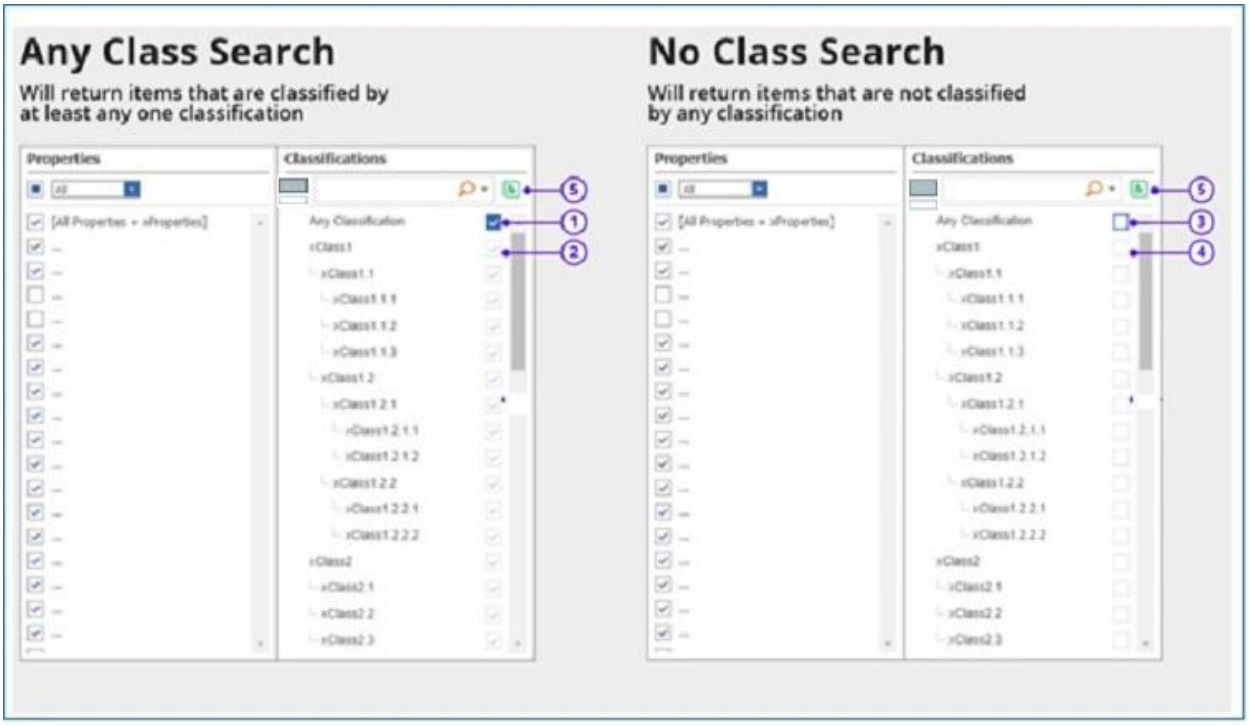

Figure 21.

1. Set “Any Classifications” to Hard Checked to search for items that are classified by at least one xClass from the xClassification Trees.
2. When you select “Any Classifications” then all the xClasses in the Trees should be Soft Checked.
3. Set “Any Classifications” to Hard Unchecked to search for items that are not included in any classification.
4. After setting “Any Classifications” to Hard Unchecked, all xClasses should display “Soft Unchecked”.

©2025 Aras Corporation All Copyrights Reserved.

The logic used has no effect on searching by “Any Classification” — if set to either “AND” or “OR” the operation functions the same.

6.1.1.3 Using The xClass Bar

When you select xClasses and click the  icon, they appear in the xClass bar that appears in the Main grid as shown in figure 22.

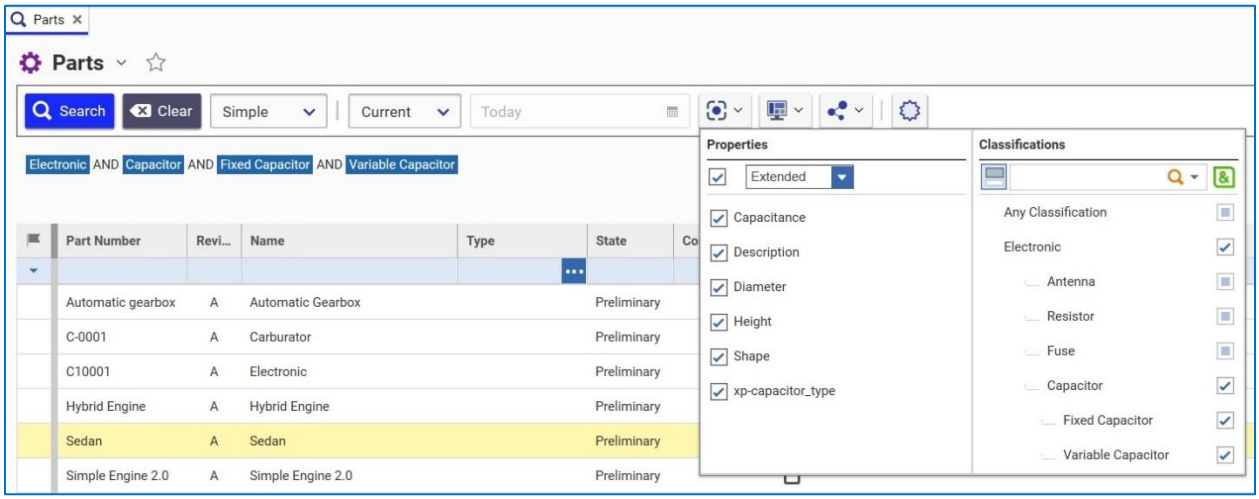

Figure 22.

The Relationship grid displays xClasses and xProperties this way:

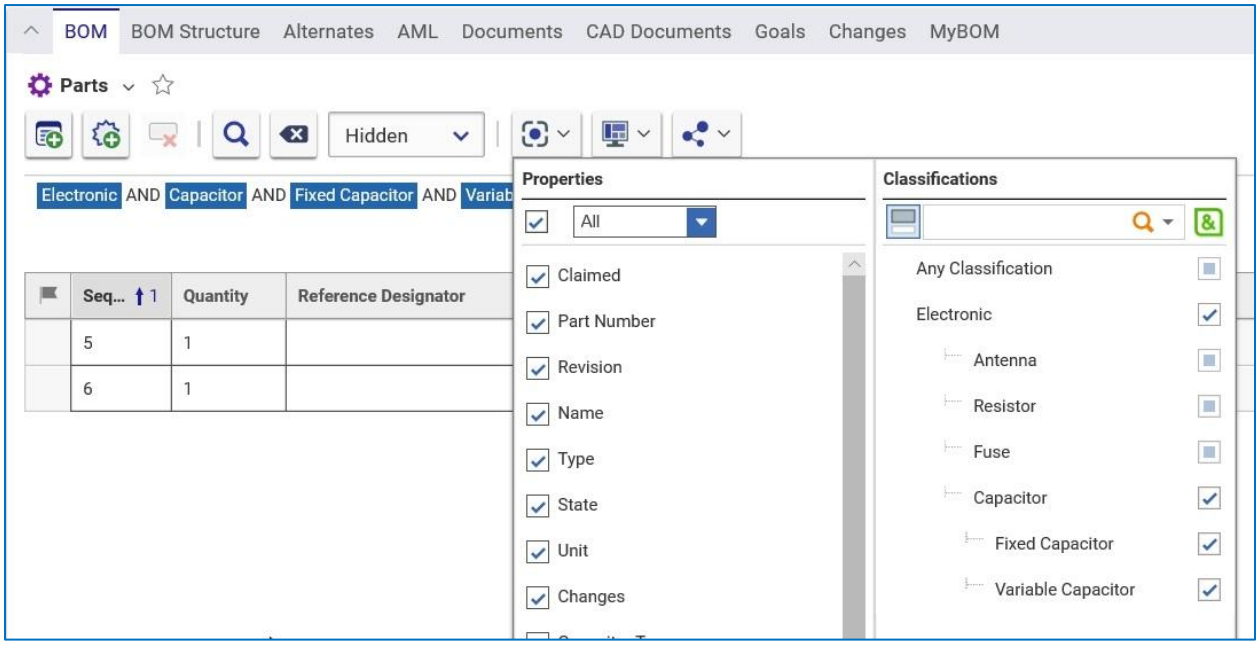

Figure 23.

©2025 Aras Corporation All Copyrights Reserved.

When you perform an Advanced search, the grid displays xClasses like this:

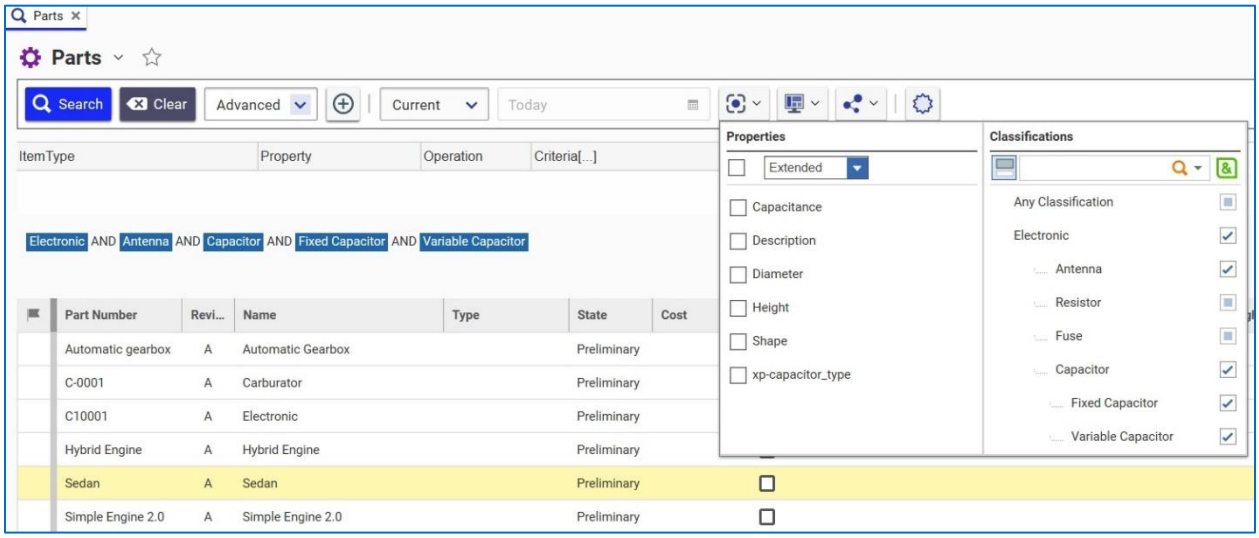

Figure 24.

For an AML search, xClasses appear this way:

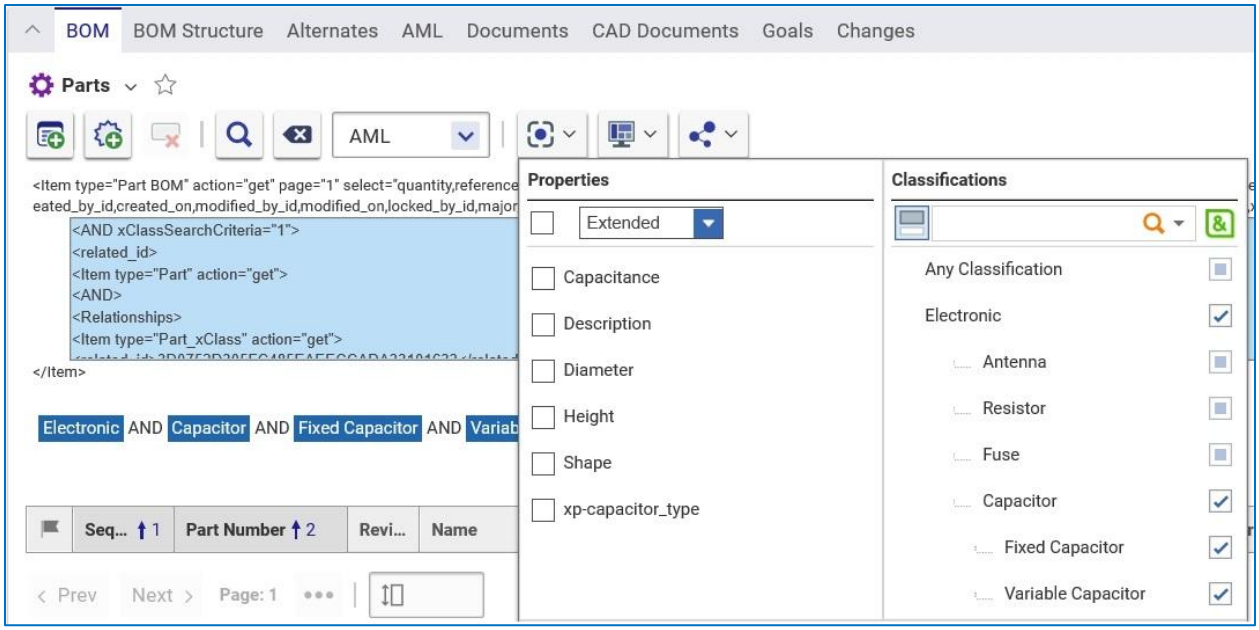

Figure 25.

©2025 Aras Corporation All Copyrights Reserved.

6.1.2 Searching for xProperties across Multiple ItemTypes

You can assign xProperties to items regardless of type. You can search for items that have xProperties associated with them using the following procedure:

1. Select **My Innovator → Extended Property Search** from the TOC. The following menu appears:

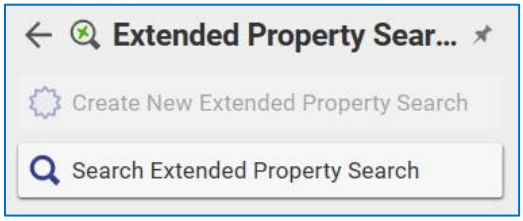

Figure 26.

2. Select **Search Extended Property Search**. The search grid appears.

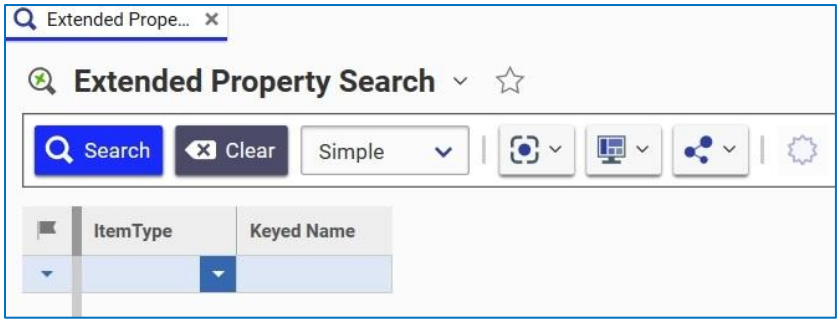

Figure 27.

3. To search for items associated with a specific ItemType, you can either enter the ItemType Name or select it from the dropdown list in the ItemType column. To see a list of all items that have xProperties associated with them, leave the ItemType column empty and click the **Search** icon.

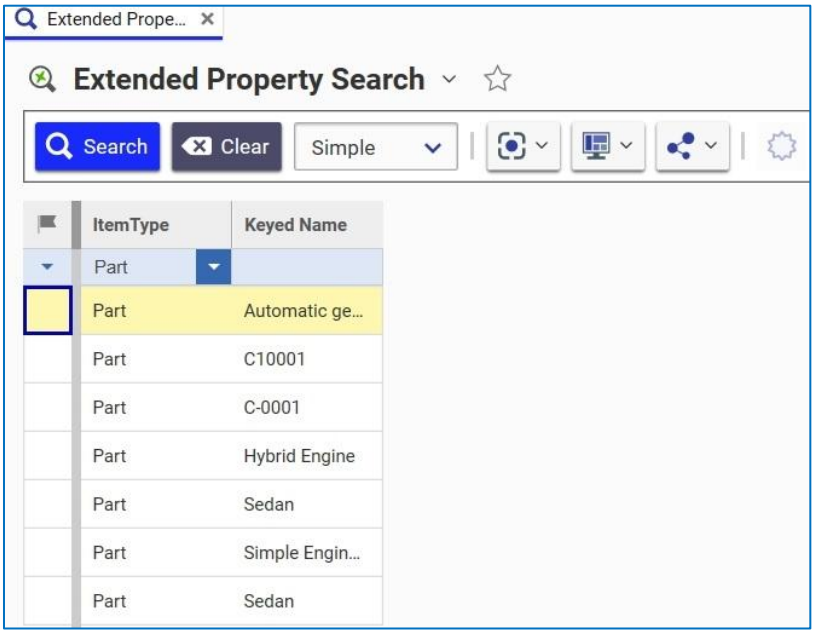

Figure 28.

©2025 Aras Corporation All Copyrights Reserved.

4. Select an item in the grid and click the **Select Columns** icon to see a list of Properties, xProperties, and xClasses associated with the selected item.

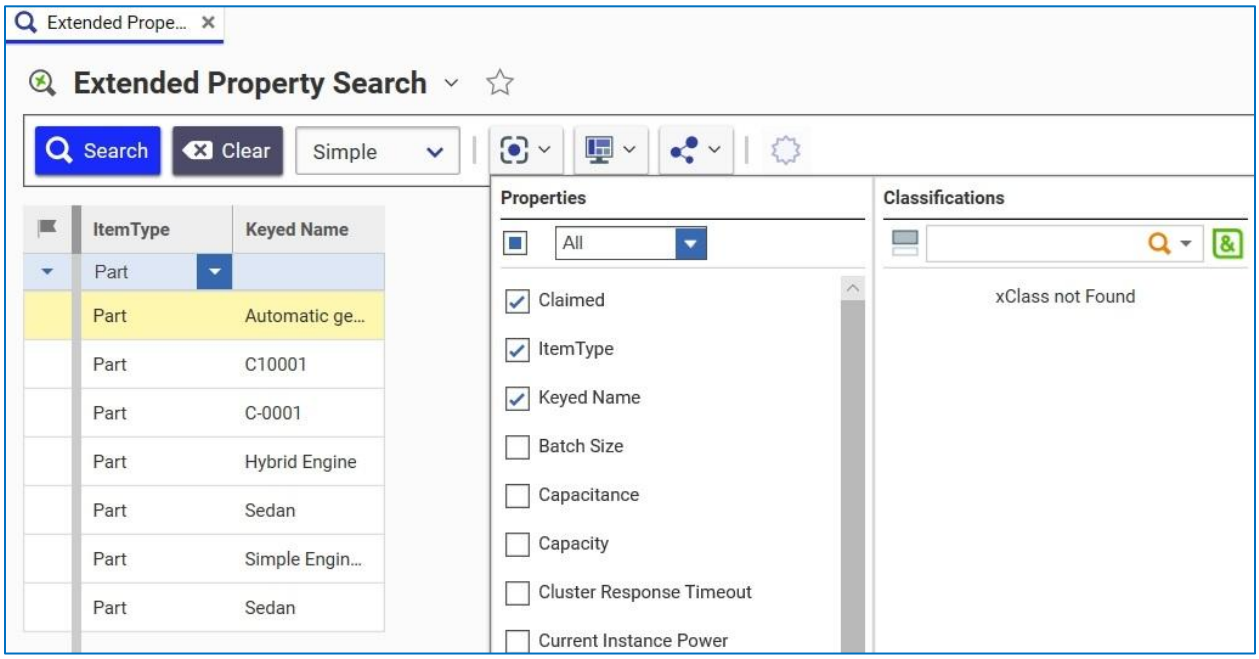

Figure 29.

5. Select **Extended** from the Properties dropdown list to see the list of extended properties associated with the item.

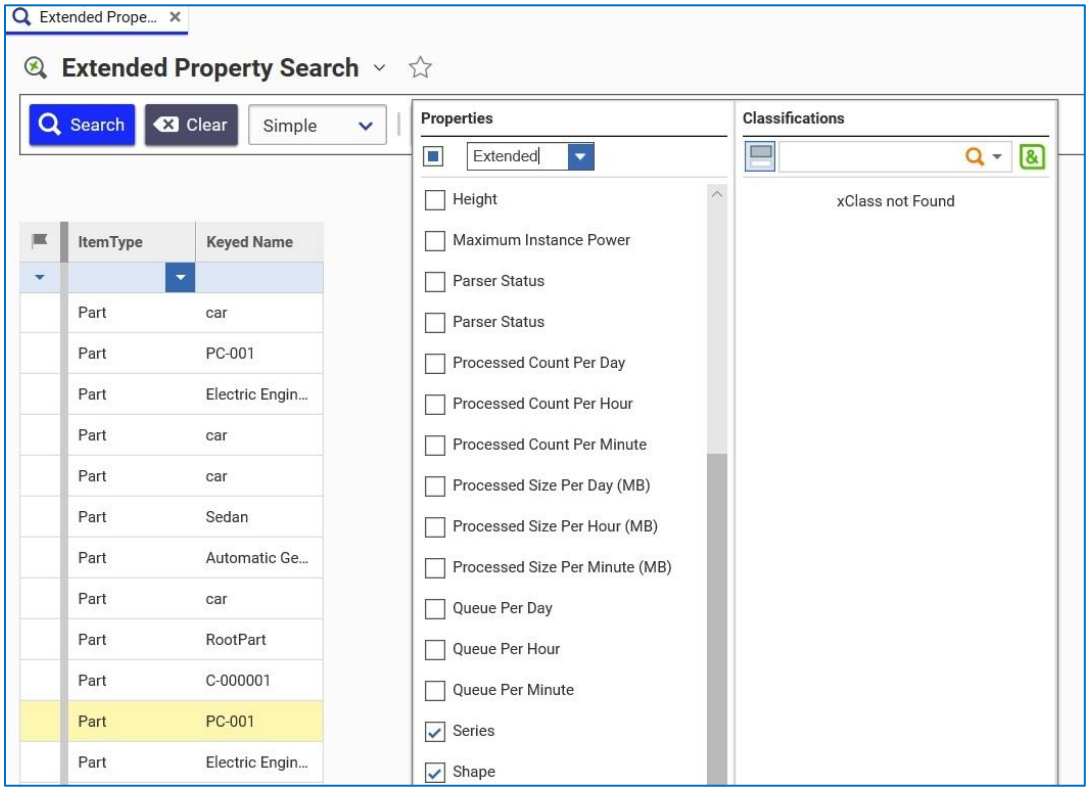

Figure 30.

©2025 Aras Corporation All Copyrights Reserved.

6.1.3 Performing an Advanced Search

You can use the Advanced Search option to search for values associated with:

- xProperties
- xClasses
- Item Classifications
- Explicit Permissions

The following procedure uses xClassification Tree as an example:

1. Select **Extended Classification > xClassification Trees** from the TOC. Clicking the Search icon takes you directly to the Search grid. Clicking xClassification Trees takes you to the following menu:

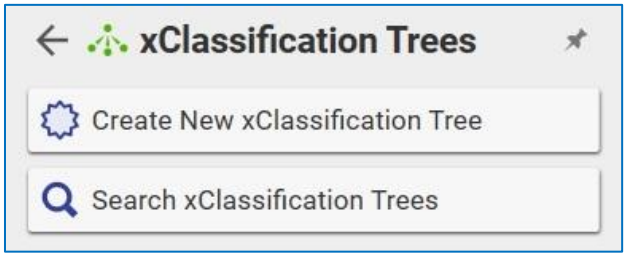

Figure 31.

2. Click **Search xClassification Trees**. The Search Grid appears.

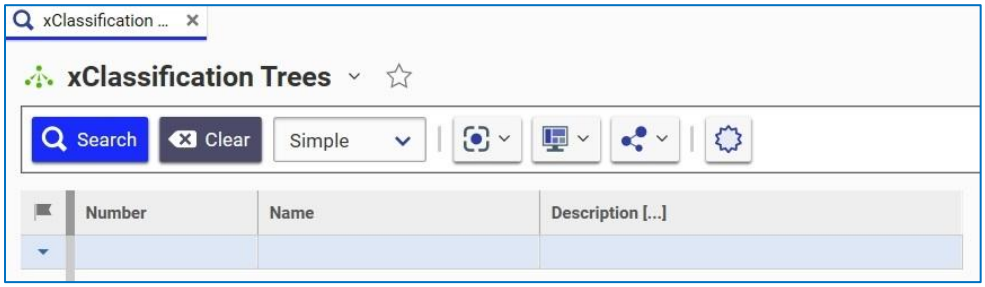

Figure 32.

3. Select **Advanced** from the Search dropdown. The xProperty Selector row appears, enabling you to search for xProperties associated with specific ItemTypes.

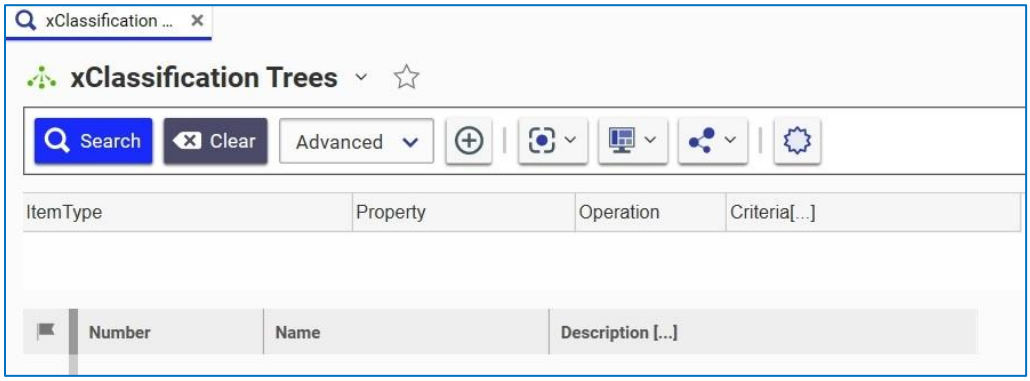

Figure 33.

©2025 Aras Corporation All Copyrights Reserved.

4. Click the **Add Criteria** icon ⊕ to add your search criteria.

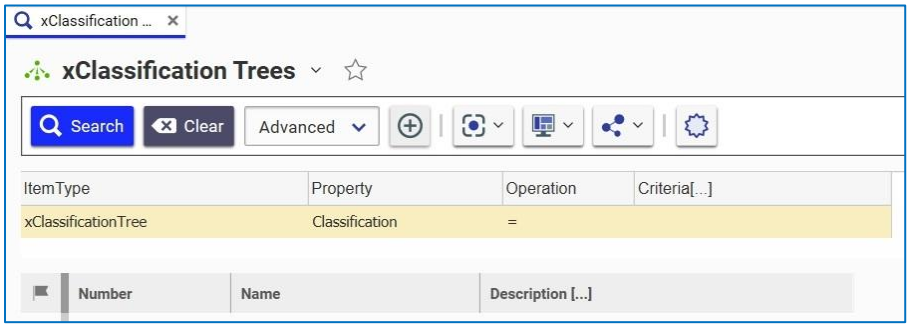

Figure 34.

5. Select the ItemType from the dropdown list:

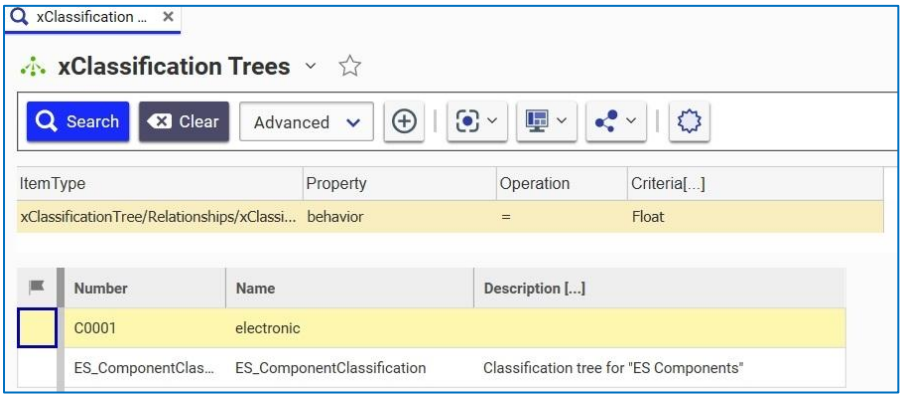

Figure 35.

6. Select the **Property** cell and click the ellipses. The Property dialog box appears.

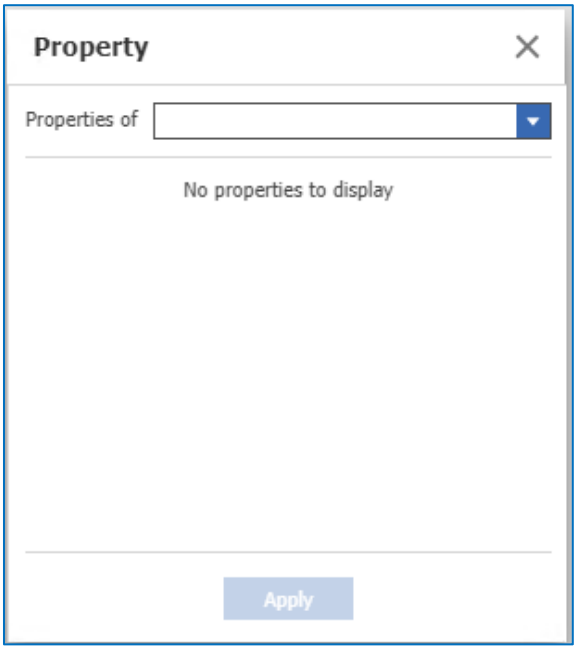

Figure 36.

©2025 Aras Corporation All Copyrights Reserved.

7. Select **xClassification Tree/Relationships/xClassificationTree_ItemType** from the Properties list. A list of properties associated with the ItemType appears.

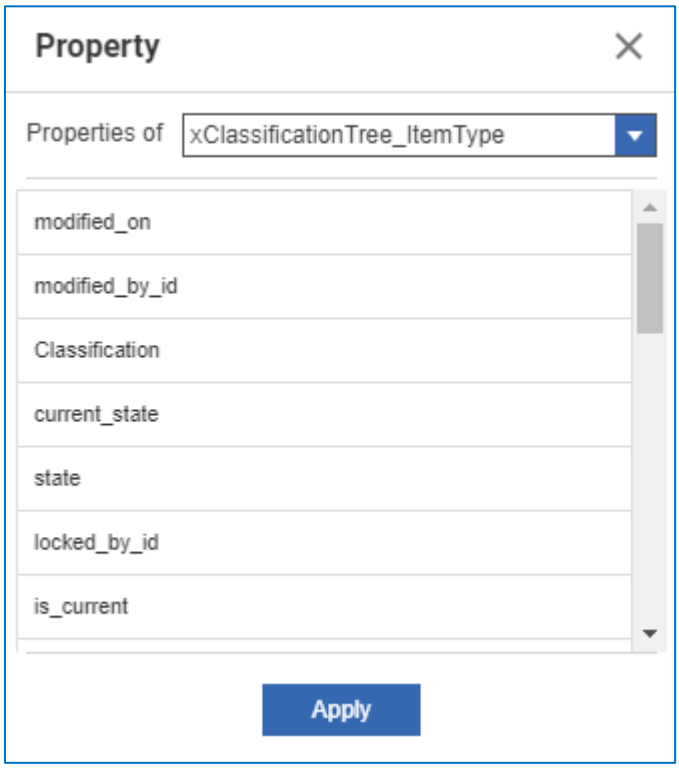

Figure 37.

8. Select **behavior** and click **Apply**.
9. Click the dropdown button in the **Operations** cell and select **=.**
10. Select the dropdown button in the **Criteria** cell and select **Float**.
11. Click **Search**. A list of xClassification trees associated with the specified criteria appears in the grid.

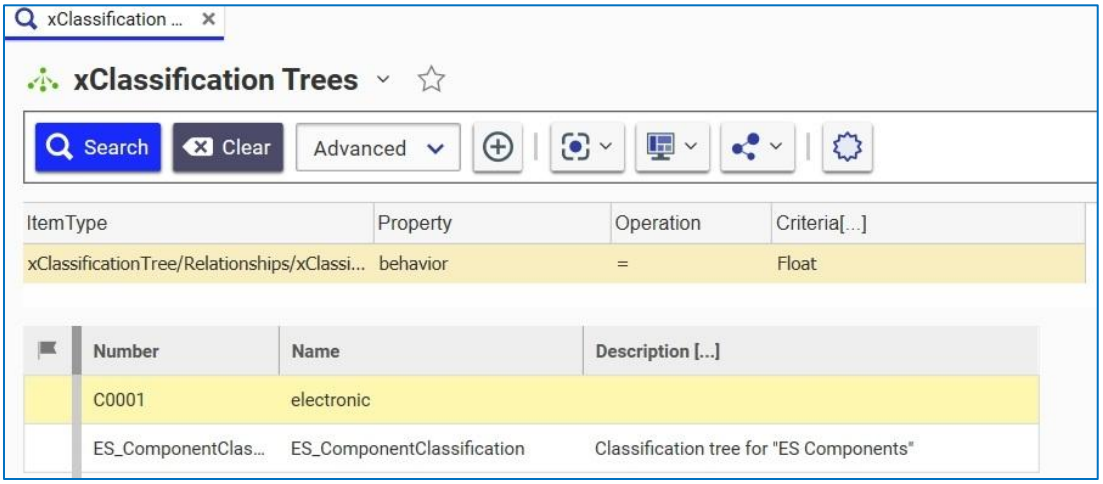

Figure 38.

©2025 Aras Corporation All Copyrights Reserved.

7 xProperties in AML

This section describes how to define attributes and xProperties programmatically in AML.

7.1 Using the @set attribute

You can perform the following operations on an xProperty when you are updating or adding an Item:

- Set a value.
- Explicitly define it
- Change the private permissions

You must add the @set attribute to an xProperty node in order to specify the operation to be performed. The following is the list of valid values associated with the @set attribute:

- “value”
- “explicit”
- “permission_id”
- Any combination of the values listed here, using the “|” as the divider (for example, “explicit|value” means “perform both of these operations”).

If you do not include the @set attribute, update operations will not occur.

7.2 Explicitly Defining an xProperty on an Item

The difference between xProperties and standard properties is that you have to define an xProperty for an item before you can perform an operation (for example get/set value) with it.

You can either define the xProperty explicitly or implicitly. To explicitly define an xProperty you have to add the @explicit and @set attributes to the Property node. If you do not specify the @set attribute or if it does not contain the value “explicit”, the “@explicit” attribute is ignored.

The following example shows how to add an item and define an xProperty programmatically. The default value is set for the “xp-cost” property because the “set” attribute does not contain a “value” string:

```
<Item type="Part" action="add">
    <xp-cost set="explicit" explicit="1"/>
    <cost>128</cost>
</Item>
```

The following example updates an item and defines the xProperty:

```
<Item type="Part" action="update" id="0FA8ED...">
    <xp-cost set="explicit" explicit="1"/>
    <cost>128</cost>
</Item>
```

This example updates the item, defines an xProperty and sets its value:

```
<Item type="Part" action="update" id="0FA8ED...">
    <xp-cost set="value|explicit" explicit="1">100<xp-cost/>
    <cost>128</cost>
</Item>
```

©2025 Aras Corporation All Copyrights Reserved.

This example updates the item and sets the value for an already defined xProperty:

```
<Item type="Part" action="update" id="0FA8ED...">
    <xp-cost set="value">100<xp-cost/>
    <cost>128</cost>
</Item>
```

This is an example of making an explicitly defined xProperty undefined:

```
<Item type="Part" action="update" id="0FA8ED...">
    <xp-cost set="explicit" explicit="0"/>
</Item>
```

7.3 Resolving Ambiguous Property Names

Use the “xp-“ prefix to distinguish an xProperty from a standard property in AML, For example, the xProperty name for the cost property would be xp-cost. This technique guarantees that names are unique because standard properties cannot use a hyphen as part of a name.

Note:	You cannot use Namespaces resolve ambiguous property names because they are used in AML to return multilingual strings. It does not work for an xProperty with a multilingual String data type. Refer to the following example. The following AML is invalid.

```
<Item xmlns:xProp="http://www.aras.com/xProperties">
    <mls>String</mls>
    <i18n:mls xml:lang="ru">Строка</i18n:mls>
    <xProp:mls>Other String</xProp:mls>
    <i18n:xPropr:mls xml:lang="ru">Другая Строка</i18n:xPropr:mls>
</Item>
```

7.4 Assigning an xPropertyDefinition to an ItemType

When you assign an xProperty to an ItemType, it is referenced in AML as a property of the ItemType. If you do not define an xProperty on an item, it is ignored. The following example gets all the items from the database where weight is equal to 15 and the xProperty cost is equal to 10:

```
<Item type="Part" action="get" select="item_number, cost, xp-cost">
    <xp-cost>10</xp-cost>
    <weight>15</weight>
</Item>

<Result>
    <Item type="Part" typeId="3B135425…" id="0FA8ED…">
        <item_number>999-888</item_number>
        <cost>127</cost>
        <xp-cost>10</xp-cost>
    </Item>
</Result>
```

©2025 Aras Corporation All Copyrights Reserved.

Get items example if the user doesn't have permissions on the xProperty:

Get all part items from the database, where the weight is equal to 15 and the xProperty cost is equal to 10. For each found item return the item_number, cost and xProperty cost (defined explicitly).
User does NOT have "can get" permissions on the xProperty cost.

```
<Item type="Part" action="get" select="item_number, cost, xp-cost">
    <xp-cost>10</xp-cost>
    <weight>15</weight>
</Item>

<Result>
    <Item type="Part" typeId="3B135425…" id="0FA8ED…">
        <item_number>999-888</item_number>
        <cost>127</cost>
        <xp-cost is_null="0"></xp-cost>
    </Item>
</Result>
```

Get items with "select=*" example:

Get all part items from the database, where the weight is equal to 15 and the xProperty cost (defined explicitly) is equal to 10. The @select attribute is not specified. The xProperty length is defined on the resulting item and has a NULL value. For each found item return all standard properties and all defined xProperties. All properties with the value NULL are not returned.

```
<Item type="Part" action="get" select="*">
    <xp-cost>10</xp-cost>
    <weight>15</weight>
</Item>

<Result>
    <Item type="Part" typeId="3B135425…" id="0FA8ED…">
        <item_number>999-888</item_number>
        <cost>127</cost>
        ... all NOT NULL standard properties here
        <xp-cost>10</xp-cost>
        ... all NOT NULL xProperties defined on this PART item here
    </Item>
</Result>
```

Get items without "select" :

Get all part items from the database, where the weight is equal to 15 and the xProperty cost is equal to 10. The xProperty length is defined on the resulting item implicitly. For each found item return all standard properties and all defined xProperties.

```
<Item type="Part" action="get">
    <xp-cost>10</xp-cost>
    <weight>15</weight>
</Item>
```

©2025 Aras Corporation All Copyrights Reserved.

```
<Result>
    <Item type="Part" typeId="3B135425…" id="0FA8ED…">
        <item_number>999-888</item_number>
        <cost>127</cost>
        <team_id is_null="1"/>
        ... all standard properties here
        <xp-cost>10</xp-cost>
        <xp-length is_null="1"/>
        ... all xProperties defined on this PART item here
    </Item>
</Result>
```

Update item example:

Update the standard property “Cost” value to 128 and xProperty “Cost” (defined explicitly or implicitly) value to 100:

```
<Item type="Part" action="update" id="0FA8ED...">
    <xp-cost set="value">100</xp-cost>
    <cost>128</cost>
</Item>
```

7.5 Using @defined_as to Filter xProperties

The following conditions enable you to filer items according to their defined status:

- “is defined”
- “is not defined”

Use the @defined_as attribute to filter items according to how they are defined. You can use the following values with the attribute:

- “class” indicates that the xProperty is defined using classification. It does not matter whether or not the xProperty is defined explicitly.
- “explicit” indicates that the xProperty is defined explicitly. It does not matter if the xProperty is defined/not defined explicitly).
- “class|explicit” indicates that the xProperty is defined both by classification and explicitly.

The following example gets all the Parts with a defined xProperty from the database. It does not matter how the xProperty is defined:

```
<AML>
    <Item type="Part" action="get">
        <xp-cost condition="is defined"/>
    </Item>
</AML>
```

The following example gets all Parts from the database that have an explicitly-defined xProperty:

```
<AML>
    <Item type="Part" action="get">
        <xp-cost condition="is defined" defined_as="explicit"/>
    </Item>
</AML>
```

©2025 Aras Corporation All Copyrights Reserved.

The following example gets all Parts from the database, which have an xProperty that is NOT defined using classification:

```
<AML>
    <Item type="Part" action="get">
        <xp-cost condition="is not defined" defined_as="class"/>
    </Item>
</AML>
```

7.6 Getting Additional xProperty Information

Every xProperty has a complex structure that includes the following information:

- A value
- A definition
- Private permission
- Flags

Note: Due to performance considerations, the default server only returns the xProperty value in the AML response.

You can extend the syntax of the @select attribute to get additional information about the xProperties defined for an item by using the following attributes:

- $value
- @permission_id
- @explicit
- @defined_as

The following example gets all Parts contained in the database. It includes the item_number and all the defined xProperties for each returned item in the response:

```
<AML>
    <Item type=Part" action="get" select="item_number, xp-*(@defined_as)"></Item>
</AML>
```

```
<Result>
    <Item type="Part" typeId="3B135425…" id="0FA8ED…">
        <item_number>999-888</item_number>
        <xp-length defined_as="class" is_null="0"/>
    </Item>
    <Item type="Part" typeId="4B135425…" id="1FA8ED…">
        <item_number>999-777</item_number>
        <xp-length defined_as="class" is_null="0"/>
        <xp-width defined_as="class|explicit" is_null="0"/>
        <xp-height defined_as="explicit" is_null="0"/>
    </Item>
</Result>
```

The following example gets additional information about the defined xProperty (permission_id, explicit):

©2025 Aras Corporation All Copyrights Reserved.

```
<Item type="Part" action="get" select="item_number, xp-cost($value, @permission_id,
@explicit), xp-length($value, @permission_id, @explicit)">
</Item>

<Result>
    <Item type="Part" typeId="3B135425…" id="0FA8ED…">
        <item_number>999-888</item_number>
        <xp-cost explicit="1">10</xp-cost>
        <xp-length permission_id="123..." explicit="0" />10<xp-length>
    </Item>
</Result>
```

The following example gets additional information about the defined xProperty (is_defined):

```
<Item type="Part" action="get"
    select="item_number, xp-*($value, @defined_as)">
</Item>

<Result>
    <Item type="Part" typeId="3B135425…" id="0FA8ED…">
        <item_number>999-888</item_number>
        <xp-height defined_as="class|explicit"/>10<xp-height>
        <xp-cost defined_as="explicit">10</xp-cost>
        <xp-length defined_as="class"/>10<xp-length>
    </Item>
</Result>
```

7.7 Changing Private Permissions

You must add the @permission_id and @set attributes to a property node to be able to set private permissions for an xProperty. The @set attribute must contain the permission_id value.

The following example sets private permissions without changing the xProperty value:

```
<Item type="Part" action="update" id="0FA8ED...">
    <xp-cost set="permission_id" permission_id="1FBA8E6…"></xp-cost>
</Item>
```

The following example sets private permissions and changes the xProperty value:

```
<Item type="Part" action="update" id="0FA8ED...">
    <xp-cost set="permission_id|value" permission_id="1FBA8E6…">100</xp-cost>
</Item>
```

The following example sets private permissions to NULL value:

```
<Item type="Part" action="update" id="0FA8ED...">
    <xp-cost set="permission_id" permission_id=""></xp-cost>
</Item>
```

The following example specifies that the permission should not be changed without using the @set attribute:

```
<Item type="Part" action="update" id="0FA8ED...">
    <xp-cost permission_id=""></xp-cost>
</Item>
```

This AML does NOT change private permissions (because @set attribute is not added) and will NOT set value to NULL (because the absense of the @set attribute is the equivalent of “set=””).

©2025 Aras Corporation All Copyrights Reserved.

You can use the UI to change a Private Permission. In the Item Form, for the given xProperty click the button next the xProperty field to make the change:

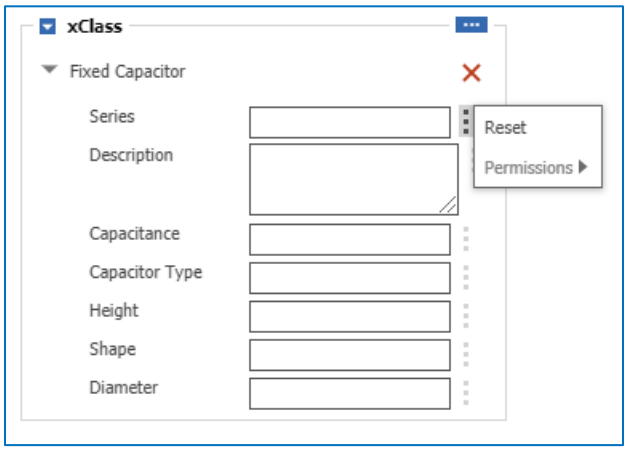

Figure 39.

7.8 Querying xProperties across Item Types

Use the xPropertyContainerItem polyitem type to retrieve values for multiple item types. Aras Innovator automatically adds the xPropertyContainerItem poly source to each item type that contains at least one allowed xProperty. The list of allowed xProperties for xPropertyContainerItem is the union of all allowed xProperties from the xPropertyContainerItem poly sources, as shown in the following example:

```
<Item type="xPropertyContainerItem" action="get" select="xp-length, xp-cost, item_id,
item_type_id">
    <xp-cost>10</xp-cost>
    <xp-weight>15</xp-weight>
</Item>
```

You can also search for xProperties across ItemTypes by selecting **My Innovator → Extended Properties Search** in the TOC. A list of ItemTypes associated with xProperties appears in the Main Grid.

7.8.1 Filtering Items by Item Type Name

You can use the xPropertyContainerItem ItemType property to filter items by item type. The @condition and @id attributes enable you to filter items by their type names, as shown in the following example:

```
<AML>
  <Item type="xPropertyContainerItem" action="get">
    <xp-cost>100</xp-cost>
    <itemtype condition="in" by="id">
      <Item type="ItemType" select="id">
       <name condition="ne">Part</name>
      </Item>
    </itemtype>
  </Item>
</AML>
```

The previous AML statement generates the following SQL:

```
SELECT … FROM [xPropertyContainerItem]
WHERE itemtype IN (SELECT id FROM secured.[ItemType] WHERE name <> 'Part')
```

©2025 Aras Corporation All Copyrights Reserved.

In this case the:

- `condition="in"` attribute tells the server that it is required to filter using a subquery.
- `by="id"` attribute indicates which property has to be used from the subquery for filtering.
- Secured function has to be used in the subquery.

7.8.1.1 Using the "condition" and "by" attributes to filter items by ANY property of ANY type

You can use the `condition="in" by="…"` attributes not only for the "itemtype" property of a poly item, but for ANY property of ANY item type.

For instance, Item Type *A* has a 'weak' reference to item type *B* not through *B.id* but through *B.name.* Item type *A* has a property *reference_to_b* which is a value of (unique) name of an item of type *B.* Now I want to use AML to find all items of type *A* that reference items of type *B* which have *cost > 10.* Then I can issue the following AML:

```
<Item type="A" action="get" select="…">
    <reference_to _b condition="in" by="name">
        <Item type="B" action="get" select="name">
            <cost condition="gt">10</cost>
        </Item>
    </reference_to_b>
</Item>
```

7.9 xProperty of Data Type Item

Xproperty type Item supports Fixed behavior only. Item Xproperties will not float to latest version of the item they reference.

7.9.1 Using $value, @keyed_name, @type in a select attribute

The following example uses the property *xp-document* of type item where the data_source is Document. The Document ItemType has the properties *name* and *description*. In order to add *name* and *description* to the output of an AML request, you have to use **$value** to get access to the *name* and *description* properties of *xp-document*.

Query:

```
<Item type="Part" select="xp-document($value(name,description))" />
```

Return:

```
<Item type="Part" id="%Part.ID%">
    <xp-document keyed_name="xp_doc_keyed_name" type="Document">
        <Item type="Document" …>
               <name>...</name>
               <description>...</description>
        </Item>
    </xp-document>
</Item>
```

If the names of properties of *xp-document* are specified without using **$value**, a null value is returned as shown here:

Query:

```
<Item type="Part" select="xp-document(@defined_as, name, description)" />
```

©2025 Aras Corporation All Copyrights Reserved.

Return:

```
<Item type="Part" id="%Part.ID%">
    <xp-document keyed_name="xp_doc_keyed_name" type="Document" defined_as="..."
is_null="0"/>
</Item>
```

Other Examples:

Query:

```
<Item type="Part" select="xp-document($value)" />
```

Return:

```
<Item type="Part" id="%Part.ID%" …>
    <xp-document keyed_name="xp_doc_keyed_name" type="Document">%Document.ID%</xp-document>
</Item>
```

Query:

```
<Item type="Part" select="xp-document($value(*))" />
```

Return:

```
<Item type="Part" id="%Part.ID%" …>
    <xp-document keyed_name="xp_doc_keyed_name" type="Document">
      <Item type="Document" …>
        ... all document STANDARD properties ...
      </Item>
    </xp-document>
</Item>
```

Query:

```
<Item type="Part" select="xp-document(@defined_as, $value(*, xp-*))" />
```

Return:

```
<Item type="Part" id="%Part.ID%" …>
    <xp-document defined_as="class" keyed_name="xp_doc_keyed_name" type="Document">
      <Item type="Document" …>
        ... all document STANDARD properties ...
        ... all document xProperties ...
      </Item>
    </xp-document>
</Item>
```

©2025 Aras Corporation All Copyrights Reserved.

8 Working with the Classification Data Model in AML

This section contains examples of working with the classification data model.

8.1 Working with classification data in the context of ItemType

The following example requests all Parts that are classified by the “Bolt” xClass.
The Response to this request does NOT have to include Parts, which are classified by child xClasses of “Bolt”.

```
<AML>
  <Item type="Part" action="get">
    <Relationships>
      <Item type="Part_xClass" action="get">
        <related_id>
          <Item type="xClass" action="get">
            <name>Bolt</name>
          </Item>
        </related_id>
      </Item>
    </Relationships>
  </Item>
</AML>
```

The following example has to include Parts, which are classified by child xClasses of “Bolt”.

```
<AML>
  <Item type="Part" action="get">
    <Relationships>
      <Item type="Part_xClass" action="get">
        <related_id>
          <Item action="getxClassAndAllDescendants">
            <name>Bolt</name>
          </Item>
        </related_id>
      </Item>
    </Relationships>
  </Item>
</AML>
```

The following example requests All Parts that are classified by the xClass named “Bolt”, where the weight is equal to 15 and the xProperty cost is equal to 10. The response HAS to include Parts, which are classified by child xClasses of “Bolt”.

```
<AML>
  <Item type="Part" action="get">
    <Relationships>
      <Item type="Part_xClass" action="get">
        <related_id>
          <Item action="getxClassAndAllDescendants">
            <name>Bolt</name>
          </Item>
        </related_id>
      </Item>
    </Relationships>
    <xp-cost>10</xp-cost>
    <weight>15</weight>
  </Item>
```

©2025 Aras Corporation All Copyrights Reserved.

```
</AML>
```

The following example requests All Parts that are classified by xClass with id “ABCD…..”.
The response HAS to include Parts, which are classified by child xClasses of xClass with id “ABCD…”.

```
<AML>
  <Item type="Part" action="get">
    <Relationships>
      <Item type="Part_xClass" action="get">
        <related_id>
          <Item action="getxClassAndAllDescendants">
            <id>ABCD.....</id>
          </Item>
        </related_id>
      </Item>
    </Relationships>
  </Item>
</AML>
```

©2025 Aras Corporation All Copyrights Reserved.

9 11.0 SP12 Extended AML Enhancements

This section contains examples of AML Enhancements released as part of SP12. You can use AML to set xProperty values and attributes such as their permission_id and explicit flag. For example:

```
<AML>
  <Item type="Part" id="816AEDE3506042C38211796A3581F4B9" action="edit">
    <xp-shape set="value">Cylindrical</xp-shape>
  </Item>
</AML>
```

Available options for the “set” attribute are “value”, “permission_id”, and “explicit”. It does not matter if the xProperty is defined or not. If you want to explicitly define the xProperty, you must use “explicit.” Use the “|” separator to set several operations at the same time:

```
<AML>
  <Item type="Part" id="816AEDE3506042C38211796A3581F4B9" action="edit">
    <xp-color set="value|permission_id|explicit"
permission_id="A518B142B9CB4013A85392AA60AB7899" explicit="1">blue</xp-color>
</Item>
</AML>
```

The xClasses set on an Item are configured as relationships in AML using the pattern *%itemtype_name%*_**xClass**. The type name is automatically defined based on the name of the ItemType specified on the xClassification Tree. This means that the xClass name for the item type "Part" would be "Part_xClass.” For the “CAD” item type the xClass name would be "CAD_xClass".

For example:

```
<Item type="Part" id="816AEDE3506042C38211796A3581F4B9" action="update">
     <Relationships>
        <Item type="Part_xClass" id="CF3E37DFE57E451CB910FDFF6B429579" action="add">
            <related_id>51CF3CA830AD49BB97CE9C3553863A76</related_id>
        </Item>
        </Relationships>
</Item>
```

9.1 Using the “Is Defined/Is Not Defined” Flags

Use the Is Define/Is Not Defined flags to filter items by their defined xProperties. The @defined_as attribute enables you to precisely specify how an xProperty is defined. The attribute uses the following allowed values:

- Class enables you to define an xProperty according to its classification. It does not matter if the xProperty is defined or not defined explicitly.
- Explicit enables you to define an xProperty explicitly. It does not matter if the xProperty is defined or defined using classification.
- Class|explicit enables you to define an xProperty both explicitly and using classification.

The following example retrieves all parts from the database where the xProperty is defined:

```
<AML>
    <Item type="Part" action="get">
        <xp-cost condition="is defined"/>
    </Item>
</AML>
```

©2025 Aras Corporation All Copyrights Reserved.

This example retrieves all parts from the database where the xProperty is explicitly defined:

```
<AML>
    <Item type="Part" action="get">
        <xp-cost condition="is defined" defined_as="explicit"/>
    </Item>
</AML>
```

This example gets all the parts from the database where the xProperty is NOT defined using classification:

```
<AML>
    <Item type="Part" action="get">
        <xp-cost condition="is not defined" defined_as="class"/>
    </Item>

</AML>
```

9.2 Requesting xProperty Information

You can use the selected attribute's extended syntax to get more information about defined xProperties. Use 'xp-*' to get all the xProperties defined for an item. You can also request additional information about xProperties by including the following special values in your request.

- Select="xp-cost" returns a value
- Select="xp-cost($value)" returns the monetary value associated with an item.
- Select="xp-cost(permission_id)" returns the permission ID attribute but does not return a value.
- Select="xp-*(@defined_as)" returns all defined xProperties but does not return a value.

The following example retrieves all Parts from the database. The response also includes the item_number and all defined xProperties:

Query:

```
<AML>
    <Item type="Part" action="get" select="item_number, xp-*(@defined_as)"></Item>
</AML>
```

Return:

```
<Result>
    <Item type="Part" typeId="3B135425…" id="0FA8ED…">
        <item_number>999-888</item_number>
        <xp-length defined_as="class" is_null="0"/>
    </Item>
    <Item type="Part" typeId="4B135425…" id="1FA8ED…">
        <item_number>999-777</item_number>
        <xp-length defined_as="class" is_null="0"/>
        <xp-width defined_as="class|explicit" is_null="0"/>
        <xp-height defined_as="explicit" is_null="0"/>
    </Item>
</Result>
```

Only the xp-length is defined on this Part. The Value is not provided because we do not provide $value in select for xp-*(…)

xp-length, xp-width and xp-height are defined on this Part

The following example shows the information request for an item's defined xProperties:

Query:

```
<Item type="Part" action="get" select="item_number, xp-cost($value, @permission_id,
@explicit), xp-length($value, @permission_id, @explicit)">
</Item>
```

©2025 Aras Corporation All Copyrights Reserved.

Return:

```
<Result>
    <Item type="Part" typeId="3B135425…" id="0FA8ED…">
        <item_number>999-888</item_number>
        <xp-cost explicit="1">10</xp-cost>
        <xp-length permission_id="123..." explicit="0" />10<xp-length>
    </Item>
</Result>
```

xp-cost is defined explicitly and does not have private permission

xp-length is not defined explicitly and has private permission (123…)

The following example shows the request for information about an item's defined xProperty:

Query:

```
<Item type="Part" action="get" select="item_number, xp-*($value, @defined_as)"/>
```

9.3 Filtering Items by Item Type Name

The following example retrieves all the items in the database that are not Part where the xProperty cost is equal to 100:

```
<AML>
  <Item type="xPropertyContainerItem" action="get">
    <xp-cost>100</xp-cost>
    <itemtype condition="in" by="id">
      <Item type="ItemType" select="id">
        <name condition="ne">Part</name>
      </Item>
    </itemtype>
  </Item>
</AML>
```

condition="in" attribute tells the server that it is required to filter using the subquery

by="value" attribute indicates which property has to be used in the subquery for filtering

9.4 Filtering Items Using the Condition= In and By Attributes

You can use the condition="in" by="…" attributes to return any property for any item type. The following example finds all Type A items that reference type B items that have a cost greater than 10 associated with them:

In the following example, the user wants to retrieve all CADs with the same name as the PART items where the cost of the PART items is equal 100.

```
<AML>
  <Item type="CAD" action="get">
    <name condition="in" by="name">
      <Item type="Part" action="get" select="name">
        <cost>100</cost>
      </Item>
    </name>
  </Item>
</AML>
```

9.4.1 Backward Compatibility for Item Data Types

You do not have to specify condition="in" by="…" attributes for properties that have Item as the data type. The following code examples are equivalent:

```
<Item type="Part" action="get">
    <created_by_id>
      <Item type="User">…
```

©2025 Aras Corporation All Copyrights Reserved.

```
<Item type="Part" action="get">
    <created_by_id condition="in" by="id">
      <Item type="User">…
```

9.4.2 Adding xProperties to any PolyItem Type

Aras Innovator does not check to see if a polyitem has an xProperty associated with it. If an xProperty exists on a polymorphic item but does not exist on the poly source, a get query returns a Null value if you request an xProperty as part of an AML query as shown in the following example:

```
<AML>
    <Item type="%AnyPolyItemType%" action="get" >
        <xp-weight condition="le">1</xp-weight>
    </Item>
</AML>
```

9.4.3 Filtering Items by xClass and Descendants

The following example is a query that returns the xClass Bolt and all of its descendants:

```
<AML>
        <Item type="Part" action="get">
                <Relationships>
                        <Item type="Part_xClass" action="get">
                                <related_id>
                                        <Item type="xClass"
                                        action="getxClassAndAllDescendants">
                                                <name>Bolt</name>
                                        </Item>
                                </related_id>
                        </Item>
                </Relationships>
        </Item>
</AML>
```

Filters by the relationship between an xClass and its descendants by specifying action="getxClassAndAllDescendants"

9.4.4 Using [<*filter_expression*>]

[<*filter_expression*>] has been added to extend the @select attribute syntax. It returns a Boolean value that determines whether or not *property* should be added to the result. Use square brackets to define <*filter_expression*> after an explicit property name or "*".

is_not_null() is the only valid filter expression used in AML.
select="*[is_not_null()]" means: add all properties into the response if property value is NOT NULL.

```
<Item type="A1" action="get" />
```

is equivalent to

```
<Item type="A1" action="get" select="*[is_not_null()]"/>
```

©2025 Aras Corporation All Copyrights Reserved.

10 xClass Search API

This section describes how to extend xClass search programmatically.

10.1 Extending the SearchMode base class

SearchMode has been extended by the following properties:

- **supportXClassSearch** enables you to use XClass Search with a specific SearchMode implementation. The default value for this property is false.
- **xClassSearchCriteriaXPath** contains the XPath that helps find xClass search criteria in the search query.

10.2 Enabling xClass Search for Custom Search Mode

It is necessary to set “supportXClassSearch” flag in the constructor of a custom search mode.

SearchMode:

JavaScript

```
function MyCustomSearch(searchContainer) {
    // This flag enables the possibility to use xClass Search with this SearchMode
    this.supportXClassSearch = true;

    // MyCustomSearch initialization code
    // Call base SearchMode constructor
    SearchMode.prototype.constructor.call(this, searchContainer, aras);
}

MyCustomSearch.prototype = new SearchMode;
// MyCustomSearch implementation
```

Note: If the xClass Search criteria is not compatible with the custom search mode a validation message appears every time you try to build a query using xClass criteria.

10.3 Extending Custom SearchMode to work with xClass Search

In order to use xClass criteria with a custom search mode without validation errors, it is necessary to extend the testAmlForCompatibility, getAml, and setAml functions of the custom search mode:

SearchMode:

JavaScript

```
function MyCustomSearch(searchContainer) {
    // This flag enables the possibility to use xClass Search with this SearchMode
    this.supportXClassSearch = true;
```

©2025 Aras Corporation All Copyrights Reserved.

```
    // MyCustomSearch initialization code
    SearchMode.prototype.constructor.call(this, searchContainer, aras);
}

MyCustomSearch.prototype = new SearchMode;
MyCustomSearch.prototype.setAml = function AMLSearchMode_setAml(searchAML) {
    let xClassCriteria;
    // call base setAml function to load searchAML into the current query item
    SearchMode.prototype.setAml.call(this, searchAML);
    // if current SearchMode supports xClassSearch
    if (this.supportXClassSearch) {
        // try to find xClassSearch criteria
        xClassCriteria =
this.currQryItem.item.selectSingleNode(this.xClassSearchCriteriaXPath);
        if (xClassCriteria) {
            // if xClassSearch criteria exists then remove it from the current query
item
            xClassCriteria = this.currQryItem.item.removeChild(xClassCriteria);
            // replace searchAML by AML without xClassSearch criteria
            searchAML = this.currQryItem.item.xml;
        }
    }
    // some business logic of custom SearchMode to populate it based on provided
searchAML
    // if xClassSearch criteria was found previously
    if (xClassCriteria) {
        // append stored xClassSearch criteria to the current query item

    this.currQryItem.item.appendChild(this.currQryItem.dom.importNode(xClassCriteria,
true));
    }
}
MyCustomSearch.prototype.getAml = function AMLSearchMode_getAml() {
    let xClassCriteria;
    // if current SearchMode supports xClassSearch
    if (this.supportXClassSearch) {
        // try to find xClassSearch criteria
        xClassCriteria =
this.currQryItem.item.selectSingleNode(this.xClassSearchCriteriaXPath);
        if (xClassCriteria) {
            // if xClassSearch criteria exists then remove it from the current query
item
            xClassCriteria = this.currQryItem.item.removeChild(xClassCriteria);
        }
    }
    // some business logic to populate current query item by search criteria for
custom SearchMode
    // if xClassSearch criteria was found previously
    if (xClassCriteria) {
        // append stored xClassSearch criteria to the current query item
        this.currQryItem.item.appendChild(xClassCriteria);
    }
}
```

©2025 Aras Corporation All Copyrights Reserved.

```
MyCustomSearch.prototype.testAmlForCompatibility = function (searchAml) {
    let searchQuery = searchAml;
    if (this.supportXClassSearch) {
        // if current SearchMode supports xClassSearch
        const queryDom = this.aras.createXMLDocument();
        queryDom.loadXML(searchAml);
        // try to find xClassSearch criteria
        const xClassCriteria =
queryDom.documentElement.selectSingleNode(this.xClassSearchCriteriaXPath);
        if (xClassCriteria) {
            // if xClassSearch criteria exists then remove it from AML which should be
validated
            queryDom.documentElement.removeChild(xClassCriteria);
            searchQuery = queryDom.documentElement.xml;
        }
    }
    return SearchMode.prototype.testAmlForCompatibility.call(this, searchQuery);
}
```

©2025 Aras Corporation All Copyrights Reserved.