Oracle® Agile Product Lifecycle Management for Process Search Extensibility Guide

Release 6.2.4.x

F58013-01

May 2022

ORACLE®

Copyrights and Trademarks

Agile Product Lifecycle Management for Process

Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface

Audience

This guide is intended for client programmers involved with integrating Oracle Agile Product Lifecycle Management for Process. Information about using Oracle Agile PLM for Process resides in application-specific user guides. Information about administering Oracle Agile PLM for Process resides in the *Oracle Agile Product Lifecycle Management for Process Administrator User Guide*.

Variability of Installations

Descriptions and illustrations of the Agile PLM for Process user interface included in this manual may not match your installation. The user interface of Agile PLM for Process applications and the features included can vary greatly depending on such variables as:

- Which applications your organization has purchased and installed
- Configuration settings that may turn features off or on
- Customization specific to your organization
- Security settings as they apply to the system and your user account

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Software Availability

Oracle Software Delivery Cloud (OSDC) provides the latest copy of the core software. Note the core software does not include all patches and hot fixes. Access OSDC at:

http://edelivery.oracle.com

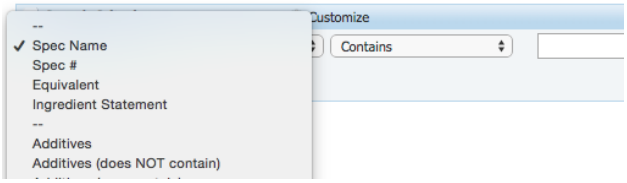
Chapter 1—Overview

While the standard application search behavior allows users to customize their experience individually, search extensibility allows you to extend this behavior. Using search extensibility you can do the following:

1.

Chapter 2—Adjusting Default Top Search Criteria

The default search criteria and the items displayed at the top of the criteria dropdown list are configurable. For example, the standard behavior displays Spec Name as the default criterion. This can be changed to any available criteria like Spec # or Equivalent.



You can make adjustments to top search criteria columns setting by editing the EQTSearchablePropertyConfigs.xml file inside the config\Extensions folder. Here are the steps:

1. Back up the EQTSearchablePropertyConfigs.xml file.
2. Find the Model node by the value of attribute modelName in this file.
3. Change the values of corresponding SearchableProperty attributes to change top search criteria.
    - To add new criteria, add a new line of SearchableProperty.
    - Remove the SearchableProperty lines to make the criteria display as normal criteria.
4. Reset IIS to review the changes in the search page.

The Model node structure in EQTSearchablePropertyConfigs looks like:

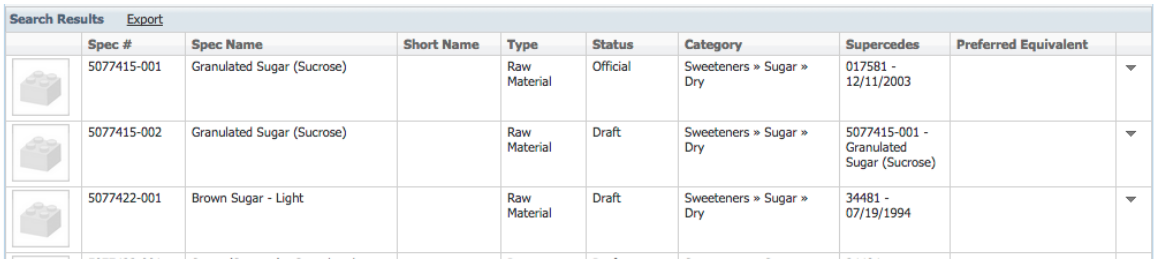
```
<Model modelName="Trade Specifications">
  <SearchableProperty propertyId="SpecName" sequence="1" enabled="true" comment=""></SearchableProperty>
  <SearchableProperty propertyId="SpecNumber" sequence="2" enabled="true" comment=""></SearchableProperty>
  <SearchableProperty propertyId="SystemEquivalent" sequence="3" enabled="true" comment=""></SearchableProperty>
  <SearchableProperty propertyId="GtinUpcNumber" sequence="4" enabled="true" comment=""></SearchableProperty>
  <SearchableProperty propertyId="OldBrand" sequence="5" enabled="true" comment=""></SearchableProperty>
  <SearchableProperty propertyId="Brand" sequence="6" enabled="true" comment=""></SearchableProperty>
</Model>
```

SearchableProperty contains following attributes:

Attribute	Value type	Description
PropertyId	string	The id of the search property. Required
sequence	integer	Sequence number of the search property, starts from 1, and the smallest on the most top. Make sure the sequence is unique. Required
enabled	boolean	true: the criteria will be on the top. false: the criteria will not be on the top. Required
comment	string	For reference only. Optional

Chapter 3—Adjusting Default Display Columns

Every search view location comes standard with default search result columns displayed. For example, the main Material Specification search shows the following columns by default.



Search Results Export

	Spec #	Spec Name	Short Name	Type	Status	Category	Supercedes	Preferred Equivalent	
	5077415-001	Granulated Sugar (Sucrose)		Raw Material	Official	Sweeteners » Sugar » Dry	017581 - 12/11/2003		
	5077415-002	Granulated Sugar (Sucrose)		Raw Material	Draft	Sweeteners » Sugar » Dry	5077415-001 - Granulated Sugar (Sucrose)		
	5077422-001	Brown Sugar - Light		Raw Material	Draft	Sweeteners » Sugar » Dry	34481 - 07/19/1994		

While individual users can configure these columns for themselves you may want to change the standard display columns to better fit the larger user community.

Default display columns are adjustable for every unique UI location using the EQTDisplayColumnsConfig.xml file inside the config\Extensions folder. You can make adjustments to display columns setting by editing the EQTDisplayColumnsConfig.xml file inside the config\Extensions folder. Here are the steps:

1. Back up the EQTDisplayColumnsConfig.xml file.

2. Find the Model node by the value of attribute modelName in this file.

3. Change the values of corresponding ColumnInfo attributes under DisplayColumns.

    - To add new display column, add a new line of ColumnInfo.

    - Remove the ColumnInfo lines to remove the display columns from search result.

    - Notice that, if there is any saved customization, you must restore to adapt the latest configuration.

4. Reset IIS to review the changes in the search page.

EQTDisplayColumns Structure

The EQTDisplayColumns node structure looks like:

```
<EQTDisplayColumns>
    <Model modelName="[MODEL_NAME]">
        <!-- Default display columns definition, required -->
        <DisplayColumns location="default">
            <!-- One or more column definition-->
            <ColumnInfo/>
            <ColumnInfo/>
            ......
        </DisplayColumns>
        <!-- Display columns definition for specific location, optional -->
        <DisplayColumns location="[LOCATION_ID]">
            <!-- One or more column definition-->
            <ColumnInfo/>
            <ColumnInfo/>
            ......
        </DisplayColumns>
        <!-- Available columns definition, required -->
        <AvailableColumns>
            <!-- One or more column definition-->
            <ColumnInfo/>
            <ColumnInfo/>
            ......
        </AvailableColumns>
    </Model>
</EQTDisplayColumns>
```

ColumnInfo under DisplayColumns contains following attributes:

Attribute	Description
location	The Location ID of EQT search which uses the display columns. The location value should be unique under a Model node. Required
orderByColumns	The sort order rule which is applied to the search results. Required
captionColumn	The index of caption column starts from 0. The caption column cannot be removed when adjusting result columns in the Customization page. Required

- Default display columns—There must be a default DisplayColumns node (**location=”default”**) under a Model node.
- Sort order rule—The format of sort order rule is defined as [COLUMN_INDEX1]|[DIRECTION1],[COLUMN_INDEX2]|[DIRECTION2],……

The columns index starts from 0, and the direction value could be ascending or descending. If the direction is not specified, it will be treated as ascending. If orderByColumns="-", no sort order rule is applied.

Example: Define different display columns for Material Specifications search

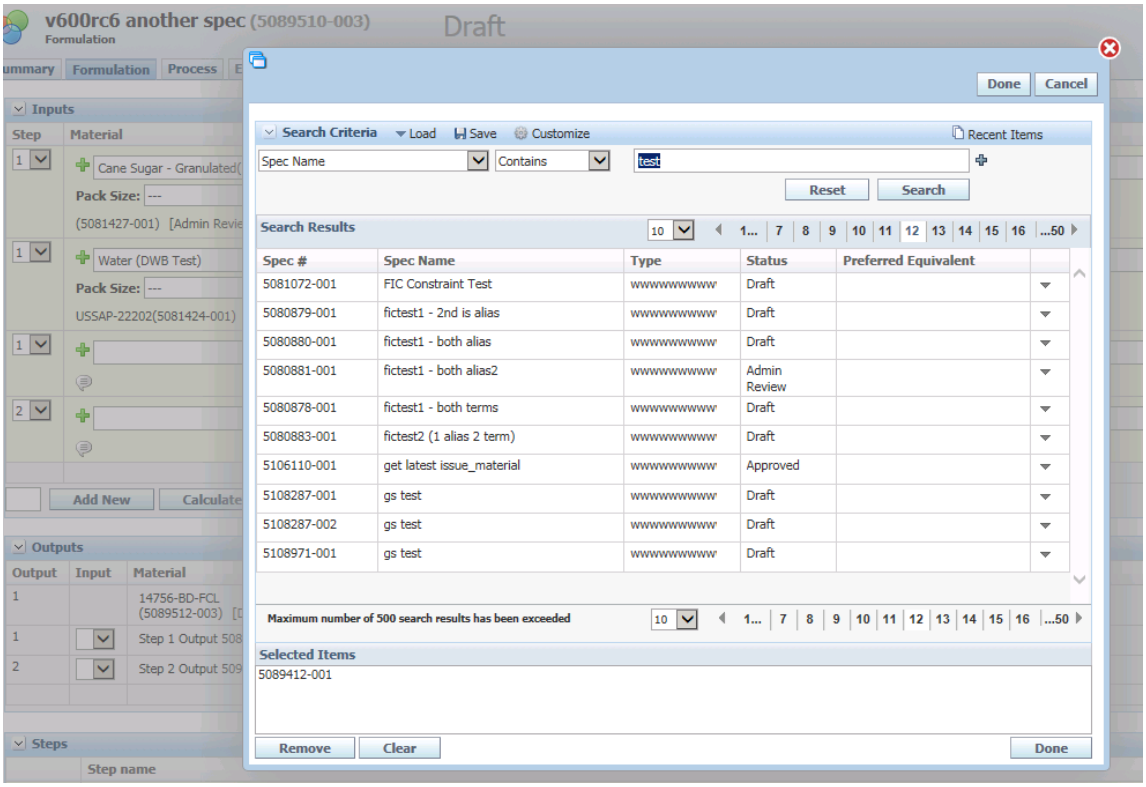
1. Find Material Specifications model node in EQTDisplayColumnsConfig.xml.
2. Add following DisplayColumns nodes. If there is already a default DisplayColumns node, remove it.

```
<DisplayColumns location="default" orderByColumns="-" captionColumn="0">
        <ColumnInfo width="79" dataField="SpecNumber"
dataFieldCaption="lblSpecNumber" provideInSelectUS="true"
isRequired="true"></ColumnInfo>
        <ColumnInfo width="178" dataField="SpecName" dataFieldCaption="lblSpecName"
isRequired="false"></ColumnInfo>
        <ColumnInfo width="75" dataField="MaterialTypeExDisplay"
dataFieldCaption="lblType"
dataTransformer="Singleton:Xeno.Prodika.EQTService.DataTransformers.MaterialTypeExDa
taTransformerFactory,PlatformExtensions" isRequired="false"></ColumnInfo>
        <ColumnInfo width="60" dataField="CurrentStatus"
dataFieldCaption="lblSpecSummaryStatus"
dataTransformer="Singleton:Xeno.Prodika.EQTService.DataTransformers.WorkflowStatusD
ataColumnTransformerFactory,EQTService" isRequired="false"></ColumnInfo>
</DisplayColumns>
<DisplayColumns location="GSM.MAT.Main"  orderByColumns="3|desc,0"
captionColumn="1">
        <ColumnInfo width="74" dataField="SpecNumber"
dataFieldCaption="lblSpecNumber" provideInSelectUS="true"
isRequired="true"></ColumnInfo>
        <ColumnInfo width="168" dataField="SpecName" dataFieldCaption="lblSpecName"
isRequired="false"></ColumnInfo>
        <ColumnInfo width="74" dataField="ShortName"
dataFieldCaption="lblSpecShortName" isRequired="false"
enableHandler="Class:Xeno.Prodika.EQTService.SearchablePropertyHandlers.FeatureProper
tyItemEnabledResolverFactory,EQTService$GSM.EQT.SpecSummary.MaterialSpec.ShortNa
me.Enabled"></ColumnInfo>
        <ColumnInfo width="58" dataField="MaterialTypeExDisplay"
dataFieldCaption="lblType"
dataTransformer="Singleton:Xeno.Prodika.EQTService.DataTransformers.MaterialTypeExDa
taTransformerFactory,PlatformExtensions" isRequired="false"></ColumnInfo>
        <ColumnInfo width="124" dataField="Taxonomy" dataFieldCaption="lblTaxonomy"
isRequired="false"></ColumnInfo>
</DisplayColumns>
```

3. Restart IIS.

Chapter 4—Configuring the Display of Selected Items in the Multi-Select Search

For the multi-select search, we are also able to choose a column as the caption column, the values of this column will be displayed in the selected items area. Let's take formulation input material search as an example. As you can see, the default caption column is Spec #.



Here are the steps of using Spec Name instead of Spec # as the caption column.

1. Find the location ID of current search. It can be found in the customization page or in the URL of the search popup. The location ID of Formulation Input Material Search is **GSM.FRM.Input**.
2. Open EQTDisplayColumnsConfig.xml inside config\Extensions\, scroll to Material Specifications.

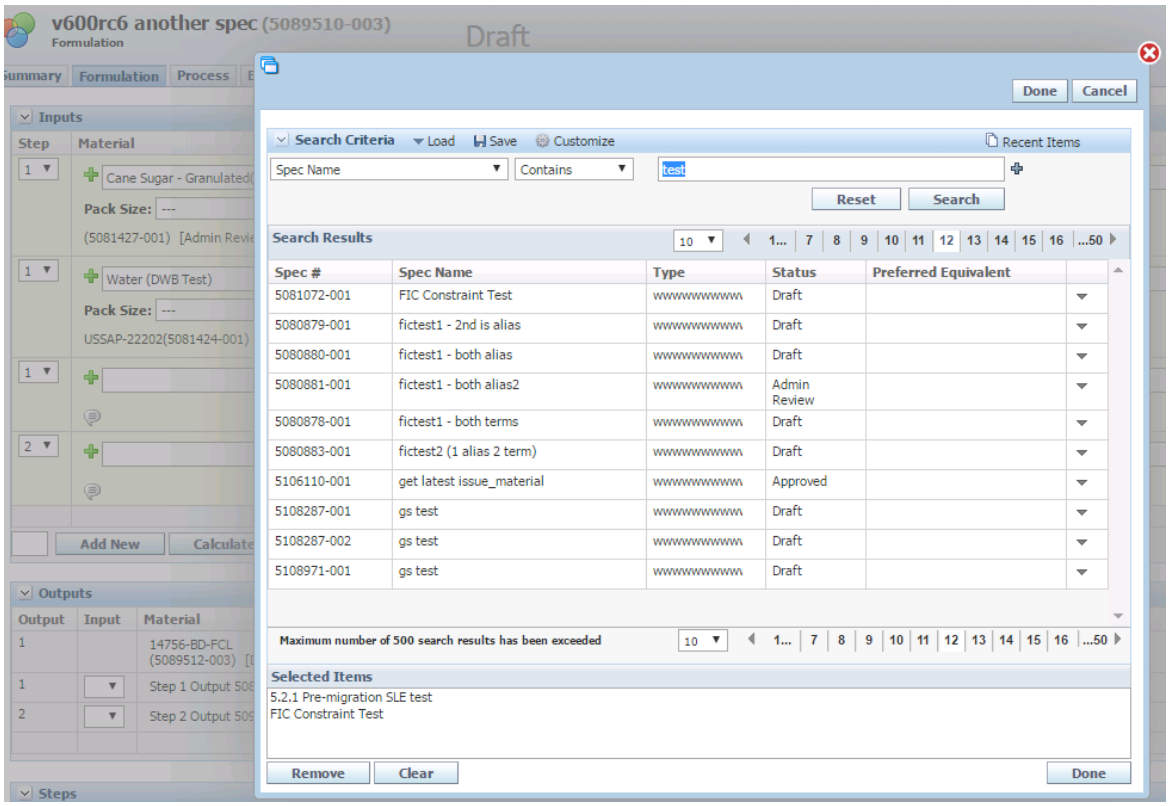
```
<Model modelName="Material Specifications">
    <DisplayColumns location="default" orderByColumns="1" captionColumn="0">
        <ColumnInfo width="79" dataField="SpecNumber" dataFieldCaption="lblSpecNumber" prc
        <ColumnInfo width="178" dataField="SpecName" dataFieldCaption="lblSpecName" isRequ
        <ColumnInfo width="75" dataField="MaterialTypeExDisplay" dataFieldCaption="lblType
        <ColumnInfo width="60" dataField="CurrentStatus" dataFieldCaption="lblSpecSummaryS
        <ColumnInfo width="136" dataField="ctxSysEquiv" dataFieldCaption="lblPreferredEqui
    </DisplayColumns>
    <DisplayColumns location="GSM.MAT.Main" orderByColumns="1" captionColumn="1">
    <DisplayColumns location="ReadyReports" orderByColumns="0" captionColumn="0">
    <AvailableColumns>
```

3. There is no DisplayColumns node for **GSM.FRM.Input**, the **default** location will be used. The captionColumn is the index of the caption column and the index starts from 0. You can change

the captionColumn to 1. This will make Spec Name as the caption column globally except location **GSM.MAT.Main** and location **ReadyReports**. Instead, you can create another DisplayColumns node for **GSM.FRM.Input** by copying the DisplayColumns node of the default location. Change the location to **GSM.FRM.Input** and the value of the captionColumn is set to **1** to display by Spec Name.

```
<Model modelName="Material Specifications">
    <DisplayColumns location="default" orderByColumns="1" captionColumn="0">
    <DisplayColumns location="GSM.FRM.Input" orderByColumns="1" captionColumn="1">
        <ColumnInfo width="79" dataField="SpecNumber" dataFieldCaption="lblSpecNumber" prov
        <ColumnInfo width="178" dataField="SpecName" dataFieldCaption="lblSpecName" isRequi
        <ColumnInfo width="75" dataField="MaterialTypeExDisplay" dataFieldCaption="lblType'
        <ColumnInfo width="60" dataField="CurrentStatus" dataFieldCaption="lblSpecSummarySt
        <ColumnInfo width="136" dataField="ctxSysEquiv" dataFieldCaption="lblPreferredEquiv
    </DisplayColumns>
    <DisplayColumns location="GSM.MAT.Main" orderByColumns="1" captionColumn="1">
    <DisplayColumns location="ReadyReports" orderByColumns="0" captionColumn="0">
    <AvailableColumns>
```

4. After saving the changes and restarting IIS, you will see the spec names in selected items.



If a customized search was saved for this location before, you must restore defaults to apply the new configuration in the customization page.

Chapter 5—Adjusting Available Display Columns

This node represents the columns which can be added to the search result. There must be one AvailableColumns node under a model node. Adjusting available display columns is mostly like adjusting display columns (see Chapter 3—Adjusting Default Display Columns) except that the available columns are under AvailableColumns.

ColumnInfo under AvailableColumns contains the following attributes:

Attribute	Description
width	This is the column width designated in pixels. Required.
dataField	The actual id for the data to be returned. Required.
dataFieldCaption	The translatable label that will be used as the column header. Required.
isRequired	Whether this column could not be removed. Required.
dataTransformer	The data transformer of the column. Optional.
dataFormatter	The data formatter of the column. Optional.
hideHeader	Whether the column header should be blank in the search result. Optional; default value is false.
sortingEnabled	Whether this column could be included in the orderBy conditions. If sortingEnabled=”false”, this column will not be listed in the Sort Order popup. Optional, default value is true.
isAggregatable	Whether data in this column can be aggregated. Optional; default value is false.
aggregationDelimiter	The delimiter between the aggregated values. The attribute is effective only if isAggregatable=”true”. Optional.
enableHandler	Determines whether this column is active. Only active columns will show up in the search result. Optional.
disableExport	Whether the column is excluded from search result during export. Optional, default value is “false”.

Chapter 6—Removing Available Search Criteria

Built-in search criteria the views of which are defined in Core cannot be removed.

For built-in search criteria the views of which are defined in EQTViews-ReadyReports.xml, EQTViews-AssociatedSpecs.xml, ReadyReports.xml, EQTViews-SupplierPortal.xml in config\Extensions folder:

1. Back up the file.
2. Find the Model node by attribute name in the file.
3. Find the SearchProperty node by attribute name under SearchableProperties.
4. Set the enableHandler value to Class:Xeno.Prodika.EQTService.PropertyHandlers.BoolPropertyHandlerFactory$true if attribute type of SearchableProperties is exclude, otherwise set the enableHandler to Class:Xeno.Prodika.EQTService.PropertyHandlers.BoolPropertyHandlerFactory$false.
5. Remove the top search criteria if the property is added to the top search criteria.
6. Restart IIS.

For criteria defined in config\Extensions\EQTModelCustomExtensions.xml:

1. Back up the file.
2. Find the ExtendEntityModel node by attribute extends in the file.
3. Find the TextProperty or NumericProperty or DateTimeProperty or PickListProperty node by attribute name.
4. Remove TableAlias nodes which are not used any more.
5. Remove the top search criteria if the property is added to the top search criteria.
6. Restart IIS.

Chapter 7—Adding New Search Criteria and Display Columns

User can create new search criteria and display columns for search model by editing the EQTModelCustomExtensions.xml located inside config\Extensions folder. Here are the steps:

1. Back up the EQTModelCustomExtensions.xml file.
2. Find the ExtendEntityModel node by attribute extends.
3. Add a TextProperty or NumericProperty or DateTimeProperty or PickListProperty.
4. Reset IIS to review the changes in the search page.

The property node name could be:

Property node name	Description
NumericProperty	Numeric property
TextProperty	Text property
DateTimeProperty	Date property
PickListProperty	Pick list, usually for searching

The property node contains following attributes:

Attribute	Description
caption	Used for display in the criteria or display column header. Required.
name	The id of the property. Optional, use caption value if not presented.
isMultiValued	Whether the pick list returns multi values, for PickListProperty only.
forSearching	Whether the property is for searching, or for display.
isTranslateable	Whether the value is translatable.
isUsePrimaryLang	Whether use primary language.
isUnicode	Whether the value is Unicode.
pickListType	Pick list type, should be S or M or H, which mean Single Column List or Multi Column List or Hierarchy.

The property node contains following sub nodes:

Sub Node name	Description
FieldName	The value field name. Should be TableName.ColumnName. Required.
JoinLinks\Links	Equation for table join, like Table1.Column1 = Table2.Column *= means left outer join =* means right outer join

Sub Node name	Description
TextCriterionOperations	Search criteria operations for TextProperty only, which contains: Starts with, Ends with, Contains, Equals, Not equals Required if for searching.
CriterionOperations	Search criteria operations for properties except TextProperty only, which contains: Greater than, Less than, Equals Required if for searching.
DataSourceName	Data source name for PickListProperty only. Required if the property is PickListProperty.

TableAlias node should be used when a table is referred to multi times.

Example: Add display column which retrieves Quantitative Range type extended attribute value to Packaging Specifications search

1. Find node <ExtendEntityModel extends="Packaging Specifications"> in EQTModelCustomExtensions.xml. Add an additional property node under the ExtendEntityModel node.

```
<TextProperty caption="lblEARoot1" name="EARoot1" forSearching="No"
isTranslateable="No" isFreeTextLanguage="No" isUnicode="No">
        <FieldName>specSummary.SpecID</FieldName>
</TextProperty>
```

2. Find node <Model modelName="Packaging Specifications"> in EQTDisplayColumnsConfig.xml. Add a ColumnInfo node like this under the AvailableColumns node. The data transformer retrieves the Quantitative Range type extended attribute value which ID is “jxc_EA_Quantitative Rang”.

```
<ColumnInfo width="115" dataField="EARoot1" dataFieldCaption="lblEARoot1"
dataTransformer="Class:Xeno.Prodika.EQTService.DataTransformers.ExtendedAttributeQu
antitativeRangeDataTransformerFactory$jxc_EA_Quantitative Rang"
isRequired="false"></ColumnInfo>
```

3. Add translation value for the data field caption.
insert into commonXLAExtensionCacheItem (pkid, fkParent, langID, Id, Value)
SELECT 'Translation PKID', '10587ed83e2e-5fd8-4eaa-958e-b854d92a7985', 0, N'lblEARoot1',
N'EA Root1 Translation' FROM DUAL WHERE NOT EXISTS(SELECT 1 FROM
commonXLAExtensionCacheItem WHERE pkid = 'Translation PKID' and langID = 0);

Verify that the Translation PKID that is being inserted is unique per language id. It must start with 1059.

4. Reset IIS and review the changes:
    a. Open the Packaging Specifications Search page, open the Add Columns dialog from the Customization page. Notice the **EA Root1 Translation** column is listed as available column.
    b. Add the **EA Root1 Translation** column and apply the changes. Run a search, notice the EA value is returned in the search result.
    c. Because the new property is not for searching, it is not listed in the Key field drop down list.

Example: Add display column which opens a Ready Report for Trade Specification search

1. Find node <ExtendEntityModel extends="Trade Specifications"> in EQTModelCustomExtensions.xml. Add an additional property node under the ExtendEntityModel node.

```
<TextProperty caption="lblReadyReportAO" name="ReadyReportAO" forSearching="No" isTranslateable="No" isFreeTextLanguage="No" isUnicode="No">
        <FieldName>specSummary.SpecID</FieldName>
</TextProperty>
```

2. Create a format plugin which implements IFormatPluginUIExtension interface like this:

```
public class OpenReadyReportPluginin : IFormatPluginUIExtension
{
        public string GetText(IFormatPluginContext context)
        {
                throw new NotImplementedException();
        }
        public string GetTextURL(IFormatPluginContext context)
        {
                const string format = "<div><a
onclick=\"YDialog.open('http://<HOST>/ReadyReports/readyreports.aspx?pkid=<READY_RE
PORT_PKID>&amp;amp;ao={0}',
'readyreports','height=750,width=950,status=no,toolbar=no,menubar=no,location=no,depe
ndent=yes,scrollbars=yes');\">Open</a></div>";
                return string.Format(format, context.Context);
        }
        public bool UseTextURL(IFormatPluginContext context)
        {
                return true;
        }
}
```

3. Put the compiled dll file under gsm\bin folder.
4. Add an additional plugin node under <FormatPlugins> node for the plugin created in step2:
   <Plugin name="EQT.OpenReadyReport" FactoryURL="Class:<CLASS_PATH>" MaxSizeUI="256" ></Plugin>
5. Find node <Model modelName="Trade Specifications"> in EQTDisplayColumnsConfig.xml. Add a ColumnInfo node like this under the AvailableColumns node. The data formatter will invoke the plugin created in step 2.

```
<ColumnInfo width="67" dataField="ReadyReportAO"
dataFieldCaption="lblReadyReportAO" isRequired="false" hideHeader="true"
dataFormatter="Class:Xeno.Web.UI.Common.Plugins.ReportLinkDelegatePlugin,WebComm
on$EQT.OpenReadyReport"></ColumnInfo>
```

6. Reset IIS and review the changes in Trade Specification search. Add the ReadyReportAO column to search result, and open the Ready Report dialog by clicking the cell in search result.

Example: Add search criteria which searches spec owner user for Material Spec

1. Find node <ExtendEntityModel extends=" Material Specifications "> in EQTModelCustomExtensions.xml. Add an additional property node under the ExtendEntityModel node.

```
<PickListProperty caption="lblSpecOwner" name="SpecOwner" forSearching="Yes" isTranslateable="No" isFreeTextLanguage="No" isUnicode="No" isMultiValued="No" pickListType="M">
        <FieldName>gsmSpecOwners.OwnerID</FieldName>
        <JoinLinks>
            <Link>specSummary.PKID *= gsmSpecOwners.fkSpecSummary</Link>
        </JoinLinks>
        <CriterionOperations equals="On"></CriterionOperations>
<DataSourceName>DataSourceURL:@APP_PATH@/WebCommon/EQTPopups/ExtendedEQTSearchPopup.aspx?MaintainSpec=true&amp;amp;CurrentValues=&amp;amp;ModelViewURL=SearchableView:Config:ProdikaSettings/EQTConfiguration/SearchableMultiSelectViews,UserViewExt&amp;amp;ScriptMethod=SelectEntity^titlebar=no,width=600,height=405,left=500,top=10</DataSourceName>
</PickListProperty>
```

2. Add translation value for the data field caption.
   insert into commonXLAExtensionCacheItem (pkid, fkParent, langID, Id, Value)
   SELECT 'Translation PKID', '10587ed83e2e-5fd8-4eaa-958e-b854d92a7985', 0, N' lblSpecOwner', N'SpecOwnerUser' FROM DUAL WHERE NOT EXISTS(SELECT 1 FROM commonXLAExtensionCacheItem WHERE pkid = 'Translation PKID' and langID = 0);
   Verify that the Translation PKID that is being inserted is unique per language id. It must start with 1059.
3. Reset IIS and review the changes in the Facility Profile search page.
   a. User can search for material specification by spec owner (only user, not group).

Example: Add search criteria and display column which represents Phone for Facility Profile search

1. Find node <ExtendEntityModel extends="Facilities"> in EQTModelCustomExtensions.xml. Add an additional property node under the ExtendEntityModel node.

```
<TextProperty caption="lblFacilityPhone" name="FacilityPhone" forSearching="Yes" isTranslateable="No" isFreeTextLanguage="No" isUnicode="Yes">
        <FieldName>
            scrmFacility.Phone
        </FieldName>
        <TextCriterionOperations startsWith="On" contains="On" equals="On"></TextCriterionOperations>
</TextProperty>
```

2. Find node <Model modelName="Facilities"> in EQTDisplayColumnsConfig.xml. Add a ColumnInfo node like this under the AvailableColumns node.

```
<ColumnInfo width="75" dataField="FacilityPhone" dataFieldCaption="lblFacilityPhone" isRequired="false"></ColumnInfo>
```

3. Add translation value for the data field caption.
   insert into commonXLAExtensionCacheItem (pkid, fkParent, langID, Id, Value)
   SELECT 'Translation PKID', '10587ed83e2e-5fd8-4eaa-958e-b854d92a7985', 0, N'lblFacilityPhone', N'Phone' FROM DUAL WHERE NOT EXISTS(SELECT 1 FROM commonXLAExtensionCacheItem WHERE pkid = 'Translation PKID' and langID = 0);
   Verify that the Translation PKID that is being inserted is unique per language id. It must start with 1059.
4. Reset IIS and review the changes in Facility Profile search page.
   a. User can search for Facility Profile by Phone.
   b. User can add Phone to the search result.

Below is an example of the Display Columns configuration for the Main Material Specification Search.

```
<DisplayColumns location="GSM.MAT.Main" orderByColumns="3|desc,0" captionColumn="0">
       <ColumnInfo width="74" dataField="SpecNumber" dataFieldCaption="lblSpecNumber"
provideInSelectJS="true" isRequired="true"></ColumnInfo>
       <ColumnInfo width="168" dataField="SpecName" dataFieldCaption="lblSpecName"
isRequired="false"></ColumnInfo>
       <ColumnInfo width="74" dataField="ShortName" dataFieldCaption="lblSpecShortName"
isRequired="false"
enableHandler="Class:Xeno.Prodika.EQTService.SearchablePropertyHandlers.FeaturePropertyItem
EnabledResolverFactory,EQTService$GSM.EQT.SpecSummary.MaterialSpec.ShortName.Enabled"></Col
umnInfo>
</DisplayColumns>
```

Available Columns

You can also remove or add columns available to users during the customize process. For example, let's say you want to add Extended Attribute "Brix" as a Display Column; however it's not something that all users will appreciate as a default column. You will want to add this column as just an Available Column to users so they can choose when to use it as a default Display Column. To adjust available columns, you follow the same approach as above however all available columns are defined in the <AvailableColumns> node below the <DisplayColumns> nodes. Available Columns are configured per Model and cannot be unique per location ID.

Chapter 8—Add New Search Criteria and Display Column

You can add new search criteria to existing search models. You can also add new display columns to existing search views. All additions are made using the EQTModelCustomExtensions.xml located inside config\Extensions folder.

The following steps should be followed:

1. Back up the EQTModelCustomExtensions.xml file.
2. Find or add the ExtendEntityModel node for the model you would like to extend.
3. Add the EQT Property Node (TextProperty, NumericProperty, DateTimeProperty or PickListProperty)
4. Add a TableAlias if the table is used more than once across a single search model.
5. If adding a column, follow the steps in "Adjusting Display and Available Columns" section to add your new column using the EQT Property you added above.
6. Reset IIS to review the changes in the search page.

The following property nodes are available:

Property node name	Description
NumericProperty	Numeric property
TextProperty	Text property
DateTimeProperty	Date property
PickListProperty	Pick list, usually for searching

Property nodes contain the following attributes:

Attribute	Description
caption	The label shown as criteria name and/or column header. You can use an extended translation or free text. Required
name	The id of the property. Required
isMultiValued	For PickListProperty Only. This determines whether the pick list returns multiple values. [Yes, No]
forSearching	Whether this property will appear as search criteria. [Yes, No]
isTranslateable	Whether the field returned is language aware. [Yes, No]

Attribute	Description
isUsePrimaryLang	Whether EQT uses the primary language for searching. The primary language is the UI Language on User Profile and Preferences. If set to false, the secondary language which is the Free Text Language on User Profile and Preferences will be used for searching.
isUnicode	Whether the field returned is Unicode. [Yes, No]
pickListType	For PickListProperty Only. Pick list type, should be S or M or H, which mean Single Column List or Multi Column List or Hierarchy
isFreeTextLanguage	Whether the translatable field returned is free text. [Yes, No]

Property nodes contain the following sub nodes:

Sub node name	Description
FieldName	The value field name. Should be TableName.ColumnName Required
JoinLinks\Links	Equation for table join, like Table1.Column1 = Table2.Column *= means left outer join =* means right outer join
TextCriterionOperations	For TextProperty Only. These are the search criteria operators: startsWith, endsWith, contains, equals, notEquals Required
CriterionOperations	These are the search criteria operators for all other property types: greaterThan, lessThan, equals Required
DataSourceName	For PickListProperty Only. This is the datasource used for the picklist. Required
ConditionalExpressions	Allows you to provide additional filters to your search query. Optional

Example: Add extended attribute as a Display Column

You are able to add numeric, quantitative range and Boolean extended attributes as display columns using the following out of the box EA Type Transformers.

Available Extended Attribute Transformers

EA Type	Transformer
Boolean	ExtendedAttributeBooleanDataTransformerFactory$EA_Boolean_ID
Numeric	ExtendedAttributeNumericDataTransformerFactory$EA_Numeric_ID
Quantitative Range	ExtendedAttributeQuantitativeRangeDataTransformerFactory$EA_Quantitative_Range_ID

Notice: For a code example about DataTransformers, you can refer to Appendix A. DataTransformer developers can customize function Object TransformCell(Object objToTransform, DataView dataView) to achieve their business requirement.

Below is an example of adding a Quantitative Range EA as a display column.

1. Find node <ExtendEntityModel extends="Packaging Specifications"> in EQTModelCustomExtensions.xml. Add an additional property node under the ExtendEntityModel node.

```
<TextProperty caption="lblEARoot1" name="EARoot1" forSearching="No" isTranslateable="No"
isFreeTextLanguage="No" isUnicode="No">
       <FieldName>specSummary.SpecID</FieldName>
</TextProperty>
```

2. Find node <Model modelName="Packaging Specifications"> in EQTDisplayColumnsConfig.xml. Add a ColumnInfo node like this under the AvailableColumns node. The data transformer retrieves the Quantitative Range type extended attribute value which ID is “EA_ID”.

<ColumnInfo width="115" dataField="EARoot1" dataFieldCaption="lblEARoot1" dataTransformer="Class:Xeno.Prodika.EQTService.DataTransformers.ExtendedAttributeQuantitativeRangeDataTransformerFactory$EA_ID" isRequired="false"></ColumnInfo>

3. Add translation value for the data field caption.

   To re-use the existing extended attribute name, use the following approach:
   insert into commonXLAExtensionCacheItem (pkid, Id, fkParent, langID, Value)
   select 'PROVIDE_1059_UNIQUE_PKID', N'PROVIDE_dataFieldCaption', '10587ed83e2e-5fd8-4eaa-958e-b854d92a7985', ml.langID, ml.Name from
   commonExtendedAttributeType tp
   inner join commonExtendedAttributeTypeML ml on tp.pkid = ml.fkExtendedAttributeType
   where AttributeID = 'EA_ATTRIBUTE_ID';

   To create a new name, use the following approach:
   insert into commonXLAExtensionCacheItem (pkid, fkParent, langID, Id, Value)
   SELECT 'PROVIDE_1059_UNIQUE_PKID', '10587ed83e2e-5fd8-4eaa-958e-b854d92a7985', 0, N'lblEARoot1', N'EA Root1 Translation' FROM DUAL WHERE NOT EXISTS(SELECT 1 FROM commonXLAExtensionCacheItem WHERE pkid = 'Translation PKID' and langID = 0);

   Note: Verify that the Translation PKID that is being inserted is unique per language id. It must start with 1059.

4. Reset IIS and review the changes:
   a. Open the Packaging Specifications Search page, open the Add Columns dialog from the Customization page. Notice the **EA Root1 Translation** column is listed as available column.
   b. Add the **EA Root1 Translation** column and apply the changes. Run a search. Notice the EA value is returned in the search result.

c. Because the new property is not for searching, it is not listed in the Key field drop down list.

Example: Add search criteria and display column which represents Phone for Facility Profile search

5. Find node <ExtendEntityModel extends="Facilities"> in EQTModelCustomExtensions.xml. Add an additional property node under the ExtendEntityModel node.

```
<TextProperty caption="lblFacilityPhone" name="FacilityPhone" forSearching="Yes"
isTranslateable="No" isFreeTextLanguage="No" isUnicode="Yes">
       <FieldName>
              scrmFacility.Phone
       </FieldName>
       <TextCriterionOperations startsWith="On" contains="On"
equals="On"></TextCriterionOperations>
</TextProperty>
```

6. Find node <Model modelName="Facilities"> in EQTDisplayColumnsConfig.xml. Add a ColumnInfo node like this under the AvailableColumns node.

```
<ColumnInfo width="75" dataField="FacilityPhone"
dataFieldCaption="lblFacilityPhone" isRequired="false"></ColumnInfo>
```

7. Add translation value for the data field caption.
   insert into commonXLAExtensionCacheItem (pkid, fkParent, langID, Id, Value)
   SELECT 'Translation PKID', '10587ed83e2e-5fd8-4eaa-958e-b854d92a7985', 0,
   N'lblFacilityPhone', N'Phone' FROM DUAL WHERE NOT EXISTS(SELECT 1 FROM commonXLAExtensionCacheItem WHERE pkid = 'Translation PKID' and langID = 0);
   Verify that the Translation PKID that is being inserted is unique per language id. It must start with 1059.
8. Reset IIS and review the changes in Facility Profile search page.
   a. User can search for Facility Profile by Phone.
   b. User can add Phone to the search result.

Appendix A—DataTransformer Code Sample

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.Data;
using Xeno.Data;
using Xeno.Data.Transformers;
using Xeno.Prodika.Application;
using Xeno.Prodika.Common;
using Xeno.Prodika.Services.UOMService;
using Xeno.Prodika.Common.Performance;


namespace Xeno.Prodika.EQTService.DataTransformers
{
    public interface IExtendedAttributeDataTransformerFactory :
IDataViewColumnCellTransformerFactory
    {
        string EA_AttributeID { get; }
    }


    public class ExtendedAttributeNumericDataTransformerFactory :
IExtendedAttributeDataTransformerFactory, ITakesParameters
       {
        private static IExtendedAttributeDataTransformerFactory m_instance = new
ExtendedAttributeNumericDataTransformerFactory();
        private string ea_AttributeID;


        public static IExtendedAttributeDataTransformerFactory GetInstance()
              {
                     return m_instance;
              }


              public IDataViewColumnCellTransformer Create( int columnIndex )
              {
            return new ExtendedAttributeNumericDataTransormer(columnIndex, EA_AttributeID);
              }


        public string EA_AttributeID
```

```
        {
            get
            {
                return ea_AttributeID;
            }
        }
        public void setParams(StringSplitter splitter)
        {
            if (splitter != null && splitter.hasMoreTokens())
                ea_AttributeID = splitter.nextToken();


        }


        private class ExtendedAttributeNumericDataTransormer : IDataViewColumnCellTransformer
               {


            protected readonly int m_columnIndex;
            protected IDictionary<string, string> m_pkidToValueMap;
            protected static readonly int k_BLOCK_SIZE = 1000;
            protected readonly string m_EAID;
            protected readonly string m_EATypePKID;


            protected static string k_SQL_PKID
            {
                get
                {
                    return @"SELECT PKID PKID from commonExtendedAttributeType a  where
a.AttributeID ='{0}'";
                }
            }
            protected virtual string k_SQL
            {
                get
                {
                    return @"select AttributeValue, fkUOM, fkRoot from
commonExtendedAttributeNumeric where fkExtendedAttributeType = '{0}' and fkRoot in ({1})";
                }
            }


            public ExtendedAttributeNumericDataTransormer(int index, string eaID)
```

```
            {
                m_columnIndex = index;
                m_EAID = eaID;
                m_EATypePKID = GetEATypePKID();
            }
            #region IDataViewColumnCellTransformer Members

            protected string GetEATypePKID()
            {
                IDataReader reader =
AppPlatformHelper.DataManager.newQuery().execute(GetEATypePKIDSQL(m_EAID));

                using (reader)
                {
                    while (reader.Read())
                    {
                        string eaTypePkID = DataReaderHelper.GetPKIDInString(reader,
"PKID").Trim();
                        return eaTypePkID;
                    }
                }
                return null;
            }
            public virtual object TransformCell(object objToTransform, DataView dataView)
            {
                string pkid = objToTransform as String;
                if (pkid == null || !XenoDataHelper.IsValidID(pkid))
                    return "";

                BuildPkidToValueMap(m_EATypePKID, dataView);

                if (!PkidToValueMap.ContainsKey(pkid))
                    return string.Empty;
                else
                    return PkidToValueMap[pkid];
            }

            protected IDictionary<string, string> PkidToValueMap
            {
```

```
                get { return m_pkidToValueMap; }
            }

            [ContainsDynamicSQL]
            protected void BuildPkidToValueMap(string pkid, DataView view)
            {
                if (m_pkidToValueMap != null)
                    return;

                m_pkidToValueMap = new Dictionary<string, string>();
                ArrayList uniquePkids = GetUniquePkidsList(view);
                int blockCount = GetBlockCount(uniquePkids.Count);


PerformanceTracker.Instance().StartTimer("ProcessExtendedAttributeNumericDataTransormerRequest
", "RequestedEATypeID_#" + m_EATypePKID);



                for (int i = 0; i < blockCount; i++)
                {
                    IList uniquePkidsSubList =
                        uniquePkids.GetRange(GetStartRange(i),
                                             GetBlockSize(i, blockCount, uniquePkids.Count));

                    IDataReader reader =
                        AppPlatformHelper.DataManager.newQuery().execute(GetSQL(pkid,
uniquePkidsSubList));

                    using (reader)
                    {
                        while (reader.Read())
                        {
                            string fkUOM = DataReaderHelper.GetPKIDInString(reader,
"fkUOM").Trim();


                            string fkRoot = DataReaderHelper.GetPKIDInString(reader,
"fkRoot").Trim();


                            string AttributeValue =
DataReaderHelper.GetStringFromReader(reader, "AttributeValue").Trim();
```

```
                            string displayUOMAbbreviation;
                            if (!StringHelper.IsStringEmpty(fkUOM))
                            {
                                displayUOMAbbreviation = UOM.FromPKID(fkUOM).Abbreviation;
                            }
                            else
                                displayUOMAbbreviation = string.Empty;



                            string displayUOMValue=string.Empty;


                            if (Convert.ToDouble(AttributeValue) !=
XenoDataHelper.NULL_PRIMITIVE_DOUBLE)
                            {
                                displayUOMValue = AttributeValue;
                            }



                            if (!StringHelper.IsStringEmpty(displayUOMValue))
                            {
                                if (PkidToValueMap.ContainsKey(fkRoot))
                                    PkidToValueMap[fkRoot] = PkidToValueMap[fkRoot] + ", " +
displayUOMValue + displayUOMAbbreviation;
                                else
                                    PkidToValueMap.Add(fkRoot, displayUOMValue +
displayUOMAbbreviation);
                            }
                        }
                    }
                }


PerformanceTracker.Instance().StopTimer("ProcessExtendedAttributeNumericDataTransormerRequest"
);
            }


            protected ArrayList GetUniquePkidsList(DataView dataView)
            {
                IDictionary uniquePkids = new HybridDictionary();
                foreach (DataRowView row in dataView)
                {
```

```
                String pkid = XenoDataHelper.RestorePKID(row.Row[ColumnIndex]) as String;

                if (pkid != null && !uniquePkids.Contains(pkid))
                {
                    uniquePkids.Add(pkid, pkid);
                }
            }
            return new ArrayList(uniquePkids.Keys);
        }

        private int ColumnIndex
        {
            get
            {
                return m_columnIndex;
            }
        }

        protected String GetEATypePKIDSQL(string eaID)
        {
            return String.Format(k_SQL_PKID, eaID);
        }

        protected String GetSQL(string pkid, IList pkidList)
        {
            return String.Format(k_SQL, pkid, StringHelper.Join(pkidList, "'", "'",
"','"));
        }

        #endregion

        protected static int GetStartRange(int blockIndex)
        {
            return blockIndex * k_BLOCK_SIZE;
        }

        protected static int GetBlockSize(int blockIndex, int totalBlocks, int itemsCount)
        {
            if (blockIndex < (totalBlocks - 1))
```

```
                    return k_BLOCK_SIZE;

                return itemsCount - GetStartRange(blockIndex);
            }

            protected static int GetBlockCount(int itemsCount)
            {
                return (int)Math.Ceiling(itemsCount / (double)k_BLOCK_SIZE);
            }
        }
       }
}
```