

Filtering a Choice List

This document provides an overview of the Custom Field type Choice List (List with Sublist based on SQL)







































Narrative

SoftwarePlanner provides several different formats for Custom Fields. Our Enterprise clients have access to the “Choice List (List with Sublist based on SQL)” that will help you filter the values available selection in a custom field.

Overview

The most common use for filtering the selections is when you have a lengthy list of items for validation in your custom field. By categorizing the values, you can provide your users with a filtered selection based on a category or grouping.

For example: We have several Product Lines for our Applications. You can create a custom List of Applications, then organize them into Product Lines. Your list might look like this:

Actions	 	ProductLine	Application
  	0 0	QA Tools	QAPlanner
  	0 0	QA Tools	QAComplete
  	0 0	QA Tools	TestComplete
  	0 0	QA Tools	LoadStarter
  	0 0	QA Tools	LoadComplete
  	0 0	Dev Tools	DevPlanner
  	0 0	Dev Tools	DevComplete
  	0 0	Dev Tools	CodeCollaborator
  	0 0	Dev Tools	PeerReviewComplete
  	0 0	Dev Tools	AQTime Standard
  	0 0	Dev Tools	AQTime Pro
  	0 0	Life Cycle Management	ALMComplete

Notice that the values in the Application column are unique, and are categorized by Product Line.

Define a custom field in Test Cases that looks like this:

Custom Fields

Custom Fields allow you to create additional fields to your specifications. The first ten custom fields can be up to 8,000 characters in size. The remaining fields can be up to 100 characters, depending on the **Field Type**. You can also define Rich Text Fields with formatting capability by using any of the first ten custom fields and **Field Type = Text**. Although any text field larger than 100 characters is automatically Rich Text enabled, it's best to use the maximum field size of 8,000. You can turn off Rich Text and create a large, plain text field by using an underscore (**_**) in the field name. This tells the system that no matter the size of the field, formatting will be turned off and only plain text allowed.

[Return to Listing](#)

Field Number	Delete?	Field Name	Field Type	Field Size
1	✗	Application	Choice List (List with Sublist based on SQL) ▼	100
Set choice list with Sublist based on SQL				
2	✗		▼	
3	✗		▼	

Be sure to click Save when prompted.

SQL-based Choice List with Sublist - Test Cases: Application

SQL-based choice lists with sublist is used to create a relational choice lists (drop down list based on previously selected drop down values) for up to 3 total drop down selections. For example, you can create a custom field and have the choice list values populated based on the SQL results. The first drop down will be the only one populated until a selection is made. The second drop down box is populated based on the selection of the first drop box. The third drop down box (if used) is populated based on the selection of the second drop down box. Type in your SQL below, and choose your connection string. Click on **VERIFY SQL** once you are done, to run the SQL and generate a preview of the drop down boxes. *This is available to custom fields and List Manager fields that are 100 or less characters in size.*

[Back](#)

Derive Choice List with Sublist from SQL

SQL:

```
Select Field1 Code1, Field1 Desc1, Field2 Code2, Field2 Desc2
From Lists
where ListTypeCode = 'Applications'
Order by Field1, Field2
```

Enter a valid SQL statement to return a list of codes and descriptions from a table or view where the first column is the ID/Value with an alias of Code1 and the second column is the Name/Description field with an alias of Desc1 to populate the first drop down box. Repeat this format for a second drop down box, using Code2 and Desc2 for the aliases. The third drop down box is optional and will have Code3 and Desc3 as the aliases.

For Example: The following SQL would be used to populate 3 different drop down boxes:

```
SELECT DISTINCT SubDiv Code1, Community Desc1, ProjId Code2,
ProjectName Desc2, Job Code3, StreetAddress Desc3
FROM Addresses
ORDER BY Community, Job
```

Connection: Local ▼

In most cases, leave this as LOCAL (your default connection). If the field should connect to a server or database other than your default, choose that connection string here

[Verify SQL](#)

Our sample Select statement refers to Field1 and Field2. These are the actual field names as in the Lists table that you use, not the labels from your field definitions. Your list is identified in the Lists table by ListTypeCode, which is the name of your List. Our list is called “Applications”. If you are referencing some other table, you would use the actual table and field names from that table.

The syntax must be exactly as depicted: each field must be followed by “Code1” and “Desc1”, the next field followed by “Code2” and “Desc2” and the third field, if used, followed by “Code3” and “Desc3”. The process uses these to build the relationship between the fields.

Click Verify SQL to try it out:

Derive Choice List with Sublist from SQL

SQL:

```
Select Field1 Code1, Field1 Desc1, Field2 Code2, Field2 Desc2
From Lists
where ListTypeCode = 'Applications'
Order by Field1, Field2
```

Enter a valid SQL statement to return a list of codes and descriptions from a table or view where the first column is the ID/Value with an alias of Code1 and the second column is the Name/Description field with an alias of Desc1 to populate the first drop down box. Repeat this format for a second drop down box, using Code2 and Desc2 for the aliases. The third drop down box is optional and will have Code3 and Desc3 as the aliases.

For Example: The following SQL would be used to populate 3 different drop down boxes:

```
SELECT DISTINCT SubDiv Code1, Community Desc1, ProjId Code2,
ProjectName Desc2, Job Code3, StreetAddress Desc3
FROM Addresses
ORDER BY Community, Job
```

Connection: Local

In most cases, leave this as LOCAL (your default connection). If the field should connect to a server or database other than your default, choose that connection string here

Preview:

AQTime Pro

AQTime Standard

CodeCollaborator

DevComplete

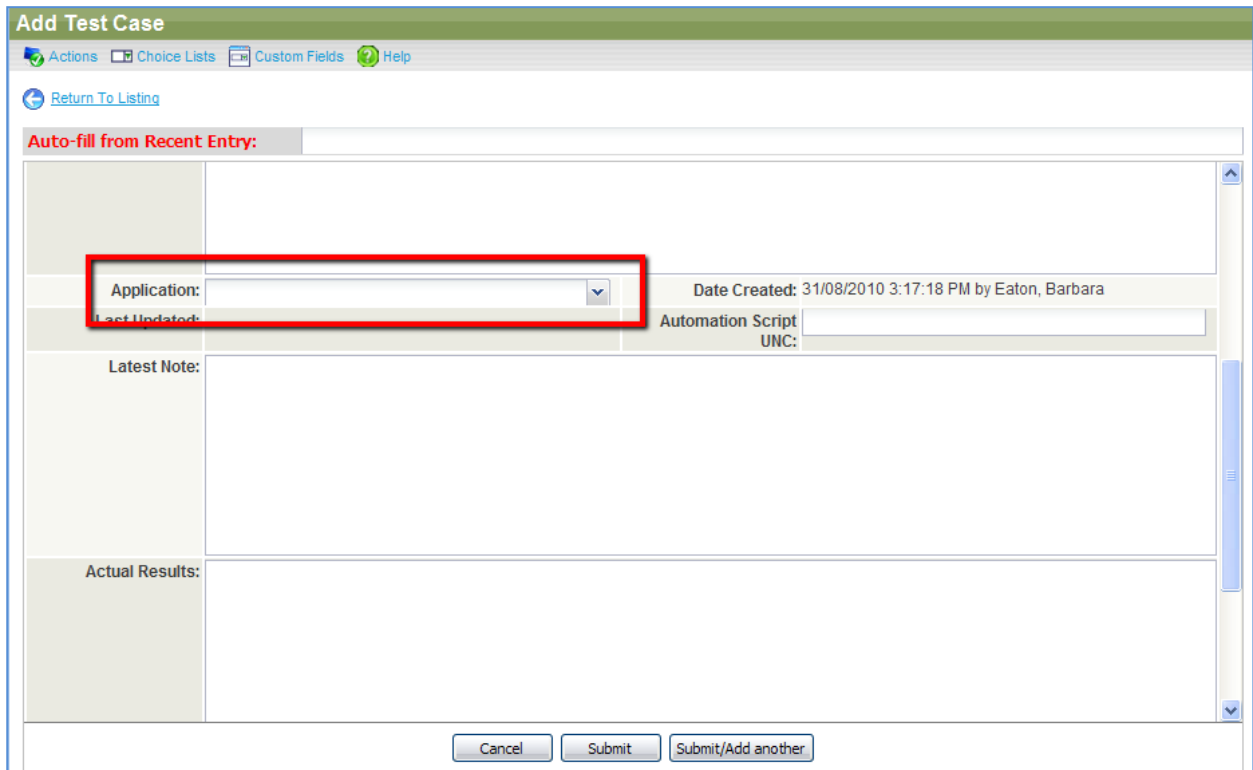
DevPlanner

PeerReviewComplete

Notice that when I select Dev Tools in the first field, the second field lists the Applications in the Dev Tools Product Line.

Click Submit.

Back to the Test Case: you'll see your Custom field for Application on the Add or Edit form.



Add Test Case

Actions Choice Lists Custom Fields Help

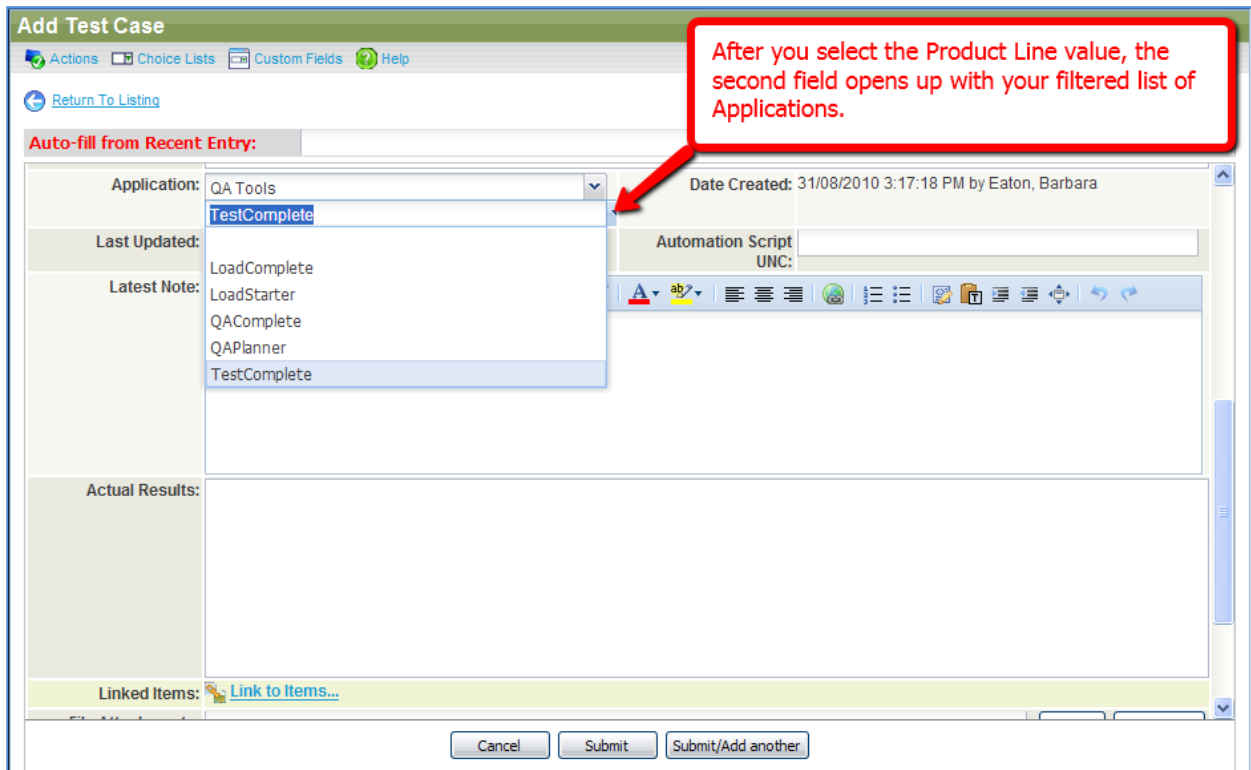
[Return To Listing](#)

Auto-fill from Recent Entry:

Application:	<input type="text"/>	Date Created: 31/08/2010 3:17:18 PM by Eaton, Barbara
Last Updated:	<input type="text"/>	Automation Script UNC: <input type="text"/>
Latest Note:	<input type="text"/>	
Actual Results:	<input type="text"/>	

Cancel Submit Submit/Add another

Click on the drop down to see your list of Product Lines. After you select a Product Line, a second field opens up.



After you select the Product Line value, the second field opens up with your filtered list of Applications.

Application: QA Tools
TestComplete

Last Updated: LoadComplete

Latest Note: LoadStarter
QAComplete
QAPlanner
TestComplete

Actual Results:

Linked Items: [Link to Items...](#)

Cancel Submit Submit/Add another

Click on the second Drop down to see the list of Applications filtered by Product Line.

Select your Application and Submit.