



University of Colorado

Boulder | Colorado Springs | Denver | Anschutz Medical Campus



OnBase Guide - Unity Script – Unity Form Repeater Concatenation

Goal: To set a property value to all/some values of a given field within a given Unity Form repeating section, using a Workflow script.

Complexity Level: Departmental Workflow Developers

2/14/2023

Table of Contents

Background	3
Examples	3
Example 1 – Getting value from all rows (no criteria)	3
Form:	3
Desired output:	4
Example 2 – Getting value from some rows (using one set of criteria)	4
Form:	4
Desired output:	4
Example 3 – Getting value from some rows (using two sets of criteria)	5
Form:	5
Desired output:	5
Prerequisites	5
Workflow Configuration	6
Set Property Values Needed for the Script	6
Required Inputs	6
Additional Inputs for Using Criteria	7
Run the Script	9
Use the Script Results	10



Background

Unity form templates can be configured to use a repeating section (or table, for purposes of this guide “repeater” refers to either type of form element) to allow users to enter multiple rows of data. For more information on repeaters, refer to the [Unity Forms MRG](#).

A repeater can be configured using a multi-instance keyword group (MIKG) or using only non-keyword fields (keyword and non-keyword fields cannot be mixed within a repeater). When using a MIKG in a repeater, workflow offers some MIKG functionality that can assist in working with the values in the repeater (though MIKG usage introduces other limitations/considerations).

For non-keyword repeaters, there are not built-in rules or actions for extracting information for use in workflow. In order to use that non-keyword repeater data, you can use the workflow Unity script as described in this guide.

This is a generic script that allows for usage from various life cycles and with various form templates/repeaters. Therefore, you will need to define some input values prior to running the script. The script can be used to concatenate values for one field from all rows in a repeater, or it can be used with additional criteria to only include rows that contain certain values.

When using workflow properties, be consistent in which property bag is being used. More information about workflow properties is available in the [Handbook](#).

Examples

Example 1 – Getting value from all rows (no criteria)

Say you have a form where students request to add one or more classes, then staff review the request and enter a decision for each class. When the form is submitted, you want to send a notification to the student that lists all the classes on their request.

Form:



Classes to Add (ID: repeatingsection_classes)	
Class Name (ID: textbox_className) SOCY 1001-100	Decision (ID: radiobuttongroup_decision) <input type="radio"/> Approved <input type="radio"/> Denied
Class Name (ID: textbox_className) EBIO 1001-300	Decision (ID: radiobuttongroup_decision) <input type="radio"/> Approved <input type="radio"/> Denied
Class Name (ID: textbox_className) MATH 1300-110	Decision (ID: radiobuttongroup_decision) <input type="radio"/> Approved <input type="radio"/> Denied

Desired output:

SOCY 1001-100, EBIO 1001-300, MATH 1300-110

Example 2 – Getting value from some rows (using one set of criteria)

With the form from the first example, you want to send another notification once staff is finished reviewing. The notification should have a list of the approved classes but not include the denied classes.

Form:

Classes to Add (ID: repeatingsection_classes)	
Class Name (ID: textbox_className) SOCY 1001-100	Decision (ID: radiobuttongroup_decision) <input checked="" type="radio"/> Approved <input type="radio"/> Denied
Class Name (ID: textbox_className) EBIO 1001-300	Decision (ID: radiobuttongroup_decision) <input checked="" type="radio"/> Approved <input type="radio"/> Denied
Class Name (ID: textbox_className) MATH 1300-110	Decision (ID: radiobuttongroup_decision) <input type="radio"/> Approved <input checked="" type="radio"/> Denied

Desired output:

SOCY 1001-100, EBIO 1001-300

Example 3 – Getting value from some rows (using two sets of criteria)

With the same form you want to send another confirmation notification once staff is finished reviewing **and** once the student is successfully enrolled. The notification should have a list of the approved classes (but not include the denied classes) and only include any that the student is enrolled in.

Form:

Classes to Add (ID: repeatingsection_classes)		
Class Name (ID: textbox_className)	Decision (ID: radiobuttongroup_decision)	Outcome (ID: radiobuttongroup_outcome)
SOCY 1001-100	<input type="radio"/> Approved <input type="radio"/> Denied	<input checked="" type="radio"/> Enrolled <input type="radio"/> Waitlisted <input type="radio"/> Denied <input type="radio"/> Pending
EBIO 1001-300	<input checked="" type="radio"/> Approved <input type="radio"/> Denied	<input type="radio"/> Enrolled <input checked="" type="radio"/> Waitlisted <input type="radio"/> Denied <input type="radio"/> Pending
MATH 1300-110	<input type="radio"/> Approved <input checked="" type="radio"/> Denied	<input type="radio"/> Enrolled <input type="radio"/> Waitlisted <input checked="" type="radio"/> Denied <input type="radio"/> Pending

Desired output:

SOCY 1001-100

Prerequisites

You must have OnBase Studio installed and know how to configure a life cycle. Refer to the [OnBase Client Guides](#) for instructions on installation and to the Workflow MRG for more details as necessary.

You'll need to have your form template configured with a repeater.

Refer to the [handbook](#) for general information on Unity script usage in Workflow.

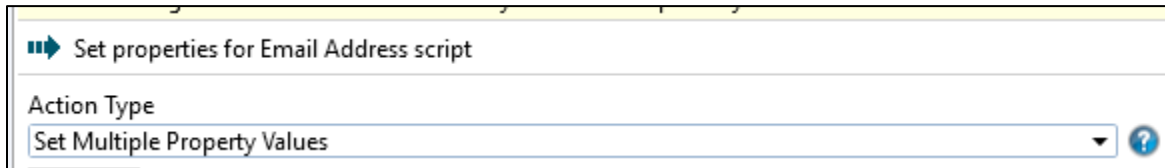
Contact UIS_DM_Support@cu.edu for assistance if needed.

Workflow Configuration

Set Property Values Needed for the Script

Required Inputs

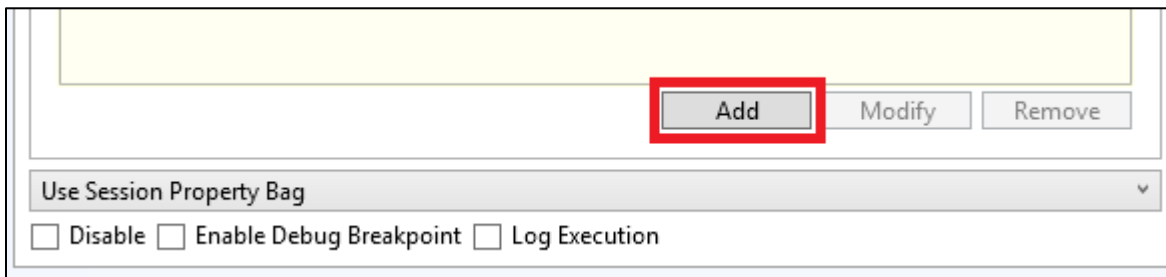
Create an action and choose the **Set Multiple Property Values** action type.



Set properties for Email Address script

Action Type
Set Multiple Property Values

Click **Add** at the bottom of the panel at the right side to add each new property value.



Add Modify Remove

Use Session Property Bag

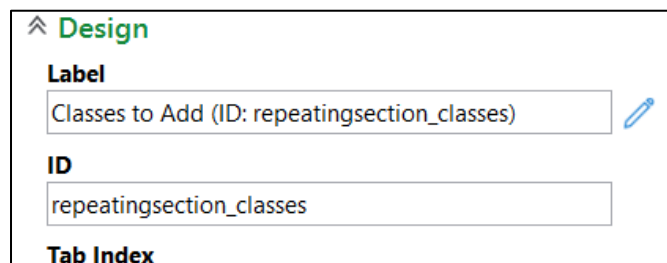
Disable Enable Debug Breakpoint Log Execution

Three inputs are required for the repeater concatenation script to run, even if no criteria are being used.

- propDelimiter
- propRepeaterFieldName
- propRepeaterName

The **propDelimiter** property should be set to the character that will separate the values from each row. For a list of email addresses, use a semicolon (;). For the examples above, you'd use a comma (,).

The **propRepeaterName** property needs to be set to the ID of the repeater on the form template. This allows the script to identify the correct repeater on any form template. For our examples, this would be *repeatingsection_classes*.



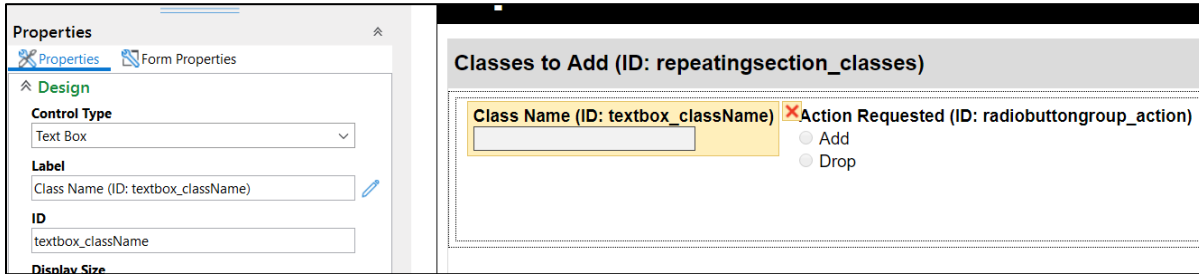
Design

Label
Classes to Add (ID: repeatingsection_classes)

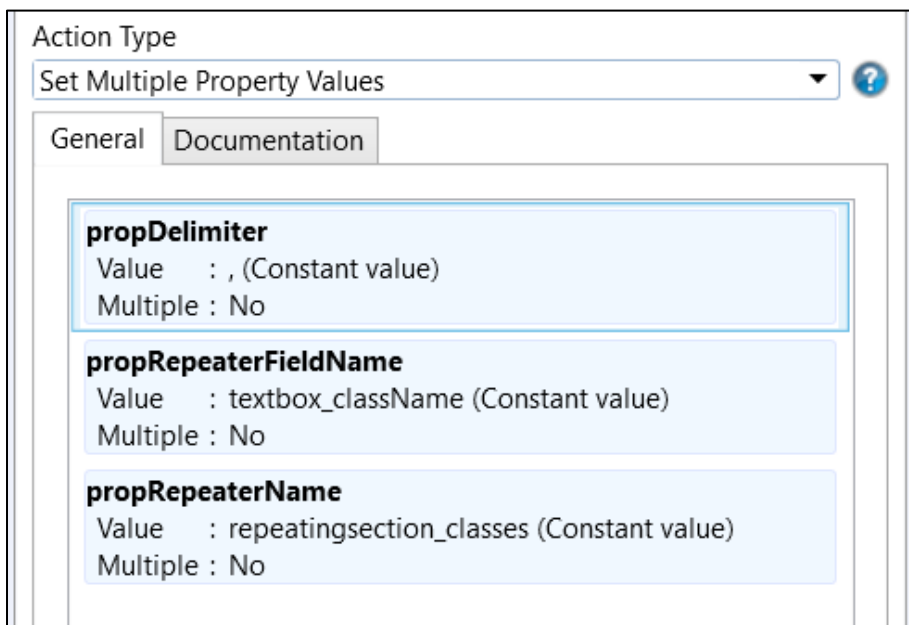
ID
repeatingsection_classes

Tab Index

The **propRepeaterFieldName** property should be set to the ID of the field in the repeater that you want to concatenate. For our examples above, we'd use *textbox_className*.



When done, the action will set all three property values:



Additional Inputs for Using Criteria

Three additional optional property values are available as inputs to add criteria to your “search.” This enables you to configure the criteria so that your output of concatenated values only includes values from repeater rows that meet all those criteria. These additional property values can be included on the same “Set Multiple Property Values” action that set the base required input property values.

The three additional criteria properties are:

- propCriteriaField
- propCriteriaValue
- propCriteriaOperator

The **propCriteriaField** property can be set to one or more field IDs. These field IDs will identify the fields to check values for to determine whether to include the value in the field referenced by **propRepeaterFieldName** in the concatenated output. If using multiple field IDs, separate them with a pipe (|) with no spaces in between.

- In the [second example](#) above (one set of criteria), since we want to only include approved classes, we would set *propCriteriaField = radiobuttongroup_decision* since we are checking the value of that field to determine whether to include the class in the list.
- In the [third example](#) above (two sets of criteria), since we want to only include approved and enrolled classes, we would set *propCriteriaField = radiobuttongroup_decision|radiobuttongroup_outcome* since we are checking the value of both fields to determine whether to include the class in the list.

The **propCriteriaValue** property needs to be set to the value(s) that correspond to the fields identified in **propCriteriaField**. If using multiple values, separate them with a pipe (|) with no spaces in between.

- In the [second example](#) above (one set of criteria), since we want to only include approved classes, we would set *propCriteriaValue = approved* since we are checking the value of the field to determine whether to include the class in the list.
- In the [third example](#) above (two sets of criteria), since we want to only include approved and enrolled classes, we would set *propCriteriaValue = approved/enrolled* since we are checking the value of both fields to determine whether to include the class in the list.

The **propCriteriaOperator** property is optional. If no input is set for this property, all criteria will be checked using “equal” as the operator. If any of your criteria are checking for “not equal”, you will need to define the operator values for each comparison. The only operators available are equal and not equal. “Not equal” should be input as *notequal* (no spaces). If using multiple values, separate them with a pipe (|) with no spaces in between.

- In the [second example](#) above (one set of criteria), since we are only checking for “equal” comparisons, the input is not needed. However we could define *propCriteriaOperator = equal*.
- In the [third example](#) above (two sets of criteria), since we are only checking for “equal” comparisons, the input is not needed. However we could define *propCriteriaOperator = equal|equal*.

When done, the action will set all six property values:

Action Type
Set Multiple Property Values

General Documentation

propCriteriaField
Value : radiobuttongroup_decision|radiobuttongroup_outcome (Constant value)
Multiple : No

propCriteriaOperator
Value : equal|equal (Constant value)
Multiple : No

propCriteriaValue
Value : approved|enrolled (Constant value)
Multiple : No

propDelimiter
Value : ,(Constant value)
Multiple : No

propRepeaterFieldName
Value : textbox_className (Constant value)
Multiple : No

propRepeaterName
Value : repeatingsection_classes (Constant value)
Multiple : No

The number of values defined in **propCriteriaValue** **must** match the number of field IDs defined in **propCriteriaField**. If using **propCriteriaOperator**, the number of operators must match as well. The two (or three) lists will be treated as corresponding sets. For example:

- propCriteriaField = fieldID1|fieldID2
- propCriteriaValue = value1|value2
- propCriteriaOperator = equal|notequal

would be interpreted as:

fieldID1 equal value1
AND
fieldID2 not equal value2

Run the Script

Create a “Run Unity Script” action. Select “GEN - OnBase - Unity Form Concatenate Repeater Fields” from the list of available scripts.

The screenshot shows a configuration window for a 'Run Unity Script' action. At the top, the 'Action Type' is set to 'Run Unity Script'. Below this, there are two tabs: 'General' and 'Documentation'. The 'General' tab is active. Under the 'Target' section, a dropdown menu is set to 'Current Item'. Under the 'Script' section, a dropdown menu is set to 'GEN - OnBase - Unity Form Concatenate Repeater Fields'. To the right of the script dropdown is a small icon with a plus sign and a pencil. Below the script dropdown is a checkbox labeled 'Refresh item after script has executed', which is currently unchecked.

When the script runs, it will output the **propConcatenatedOutput** property value with the values from the specified field (excluding any that do not meet the criteria if criteria were defined).

Use the Script Results

Once the script has set the output property value, **propConcatenatedOutput**, you can then use it in whatever rules or actions suit your process. That could mean inserting the property into a notification, copying to another form field, using the list as the list of recipients for an email, etc.