OnBase Guide - Unity Forms - Validating Field Values with RegEx

**Goal:** To ensure user-entered values are valid before form submission

**Complexity Level:** Departmental Unity Form Developers

10/11/2021
**Table of Contents**

Background ........................................................................................................................................ 3  
Using Scripts on Unity Forms ........................................................................................................... 3  
Using the GEN - OnBase - Email Address Validation Script ......................................................... 4  
  Field Invalidation ............................................................................................................................ 5  
Using the GEN - OnBase - Generalized Unity Form RegEx Search Script ................................. 7  
  Using the Script for Validation (Does the input match the given pattern?) .............................. 7  
  Using the Script to Get Output ........................................................................................................ 9
Background

Unity form fields can use data sets or masking to enforce certain data validation requirements on values entered by users. If a value does not match the mask configured for a field, the form cannot be submitted until the value is corrected. For example, to ensure a 5-digit class number is entered, you could configure this mask:

![Mask: 00000](mask.png)

For more information on masks, refer to the Unity form and System Admin MRGs.

In cases where masking or other existing custom action types are not sufficient for data validation, we have two Unity scripts available for your use:

- GEN - OnBase - Email Address Validation
- GEN - OnBase - Generalized Unity Form RegEx Search

The first can be used for email address entries, the second is a bit more complex and can be used for whatever data validation requirements you have. Both use regular expressions (regex) to provide results that are not possible with the standard set of custom actions.

For assistance, please contact UIS_DM_Support@cu.edu.

Using Scripts on Unity Forms

In order to run a script on a form, you will need to add a custom action. For more information on custom actions, please refer to the Unity Form MRG.

You will also need to make sure that the correct IDs have been assigned to the relevant fields. Form scripts identify the fields to collect information from and write information to based on the ID. The display name (label) can be whatever you like but the ID must match the script field IDs exactly. Refer to the OnBase Unity Script Inputs & Outputs Reporting Dashboard for the required inputs and outputs.

The conditions for the action will depend on your form and process, but typically one or more of these criteria are a good starting point:

- Form is new (unless you want the script to also run after forms have been submitted and are being retrieved or processed in workflow)
- Field is not empty (the value should be entered before you try to validate it)
- Condition button is clicked
In short, you will need some criteria that will be met so the form will know when to call the script. Be sure to thoroughly test this in non-production.

Then you will need an action to run the script (action type = Execute Unity Script, then select the script from the list). Other actions can also be added if necessary.

Your custom action should look something like this:

![Custom Action](image)

**Using the GEN - OnBase - Email Address Validation Script**

This script will only validate email addresses and is appropriate to use when only one email address needs to be validated and is not populated by either of the scripts to retrieve the email address from ICS or HCM based on Student ID or Employee ID. If possible, we recommend using the email address from ICS or HCM.

Please note this script will not confirm the address is an actual email account and can receive email, it is only checking that the entry could be a valid email address (contains valid address characters, then “@”, then a domain).

1. **The following fields will need to be configured on your form:**
   - `email_addr` - this is the email address being validated
   - `regexResult` - this will be populated by the script and will be set to “True” if the value could be a valid email address or “False” if it is not.

   The `regexResult` field can be hidden if desired. You can also optionally include fields to display error messages returned by the script.

2. **Configure your custom action to execute the script (as described above).**

   Example:
3. **Configure a custom action to handle the script result.**

The script will only return a true or false value to the regexResult field, it will not prevent submission of a form with an invalid address or take any other action. You will need to create one or more custom actions to achieve the desired result.

For example, if you want to require the user to enter a corrected value you will need to add some custom actions.

A simple option is to prevent submission of the form. This could also include showing a paragraph with an explanation. This solution has drawbacks for accessibility/usability.

**Field Invalidation**

If a field value is invalidated, that value cannot be used as a script input (scripts ignore values of invalidated fields). So if you mark the email_addr field invalid to prevent submission, if the user makes a correction to the value
the script cannot re-process the updated value since the field is already invalid.

A better option is to use an extra hidden field as described in this [Community post](#).

In addition to the email_addr field which will be hidden, you’ll need a field to display to users for entry. Whenever the value the user enters is changed, it will be copied to the email_addr field for the script to validate the value.

| Name: | copy email address for script check |
| Description: |  |
| **Conditions** |  |
| If [email_addr] != [email_addr display], String comparisons will be case insensitive. | And |
| If [email_addr display] is not empty | And |
| If form is new |  |

- Reverse actions when the conditions are not met.

| Actions |  |
| Set value of [email_addr] to [email_addr display]. Existing values will be overwritten. |  |

If there is a problem, the display field will be made invalid to prevent form submission.

| Name: | invalid email address |
| Description: |  |
| **Conditions** |  |
| If [regexResult] = "False", String comparisons will be case insensitive. |  |

- Reverse actions when the conditions are not met.

| Actions |  |
| Invalidate [email_addr display], with the message 'The field is invalid'. |  |
Using the GEN - OnBase - Generalized Unity Form RegEx Search Script

This script can be used to match any kind of user-input value against any regular expression pattern. This could include:

- Validating an email address if more than one email address exists on the form or if you want to ensure a certain type of email address is entered (ex. ___@cu.edu). The screenshots in the validation example below are based on this scenario.
- Checking a field entry using a wildcard-style comparison.
- This script can also be used to get output from a regex pattern instead of simply validating the value matches the given pattern, so it greatly enhances the existing custom action functionality.

This script requires additional fields on the form to tell the script what field to use and the regex pattern to use. You will need to supply the regex pattern. There are a variety of resources available for learning regex or common patterns that you can use. This list is not comprehensive but should help you get started:

- https://regex101.com/
- https://emailregex.com/

Using the Script for Validation (Does the input match the given pattern?)

These steps are for using this script only for validation of input, not using output from the regex pattern.

1. The following fields will need to be configured on your form:
   - FieldToSearch - this is what tells the script which value to validate. Set a default value on this field with the ID of the field containing the value to validate.
   - regexPattern - this is the regex pattern used to validate the value in the field specified by FieldToSearch. Set a default value on this field with the pattern you want to use. Do not include quotes around the pattern.
   - regexResult - this will be populated by the script and will be set to “True” if the value matches the provided pattern or “False” if it does not.

These fields can all be hidden (using custom actions) if desired. You can also optionally include fields to display error messages returned by the script.

Here is an example for validating a secondary email address not filled by an ICS or HCM script (here the labels on the fields are the same as the ID):
Note: The pattern shown here is simpler than the one used in the dedicated email address validation script and therefore less accurate.

2. Configure your custom action to execute the script (as described above).

Example:

3. Configure a custom action to handle the script result.

The script will only return a true or false value to the regexResult field, it will not prevent submission of a form with an invalid address or take any other action. You will need to create one or more custom actions to achieve the desired result.
For example, if you want to require the user to enter a corrected value you will need to add some custom actions.

**NOTE:** If a field value is invalidated, that value cannot be used as a script input (scripts ignore values of invalidated fields).

A simple option is to prevent submission of the form. This could also include showing a paragraph with an explanation. This solution has drawbacks for accessibility/usability.

![Custom Action](image)

A better option is to use a hidden field as described in this [Community post](#).

If you have multiple values to use with this script, you could create additional custom actions to update the values of the FieldToSearch and regexPattern fields and re-execute the script for subsequent values.

**Using the Script to Get Output**

These steps are for using this script in order to return output for the portion of the value that matches the regex pattern.

In this basic example, an academic program is entered in a form field. Based on the last letter of the program, we will display a message indicating whether it is a graduate program. This should give you an idea of the possibilities for how the script could be used and how to configure it.

1. **The following fields will need to be configured on your form:**
   - **FieldToSearch** - this is what tells the script which value to validate. Set a default value on this field with the ID of the field containing the value to validate.
   - **regexPattern** - this is the regex pattern used to validate the value in the field specified by FieldToSearch. Set a default value on this field with the pattern you want to use. Do not include quotes around the pattern.
   - **regexResult** - this will be populated by the script and will be set to “True” if the value matches the provided pattern or “False” if it does not.
• regexOutputValue - this will be populated with the output from the regex pattern if the script evaluates to True.

These fields can all be hidden (using custom actions) if desired. You can also optionally include fields to display error messages returned by the script.

Here is an example to return only the last letter of the value entered in the program field (here the labels on the fields are the same as the ID):

2. Configure your custom action to execute the script (as described above).

Example:
3. Configure a custom action to handle the script result.

The script will only return a true or false value to the regexResult field and any matching string to the regexOutputResult field. You will likely need to create one or more custom actions to achieve the desired result.

In this example, when the last letter of the program code is “G” we display a message that the program is a graduate program.
If you have multiple values to use with this script, you could create additional custom actions to update the values of the FieldToSearch and regexPattern fields and re-execute the script for subsequent values.