OnBase Guide – Workflow – FormPop Notifications

**Goal:** To configure workflow notifications using FormPop tokens to provide the recipient with a link to a Unity form

**Complexity Level:** Departmental Workflow Developers, Departmental Administrative Users

4/22/2022
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background</td>
<td>3</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>3</td>
</tr>
<tr>
<td>Resources</td>
<td>4</td>
</tr>
<tr>
<td>OnBase Configuration Client - Add Document Type to User Group</td>
<td>4</td>
</tr>
<tr>
<td>OnBase Studio - Create the Workflow Notification</td>
<td>5</td>
</tr>
<tr>
<td>Testing</td>
<td>13</td>
</tr>
<tr>
<td>Securing FormPop Content After Usage</td>
<td>13</td>
</tr>
<tr>
<td>Custom Action Recommendations to Control Visibility/Editing</td>
<td>14</td>
</tr>
<tr>
<td>Verifying Identity</td>
<td>14</td>
</tr>
<tr>
<td>Form Submission</td>
<td>15</td>
</tr>
<tr>
<td>Form is No Longer Available</td>
<td>16</td>
</tr>
<tr>
<td>FormPop and Document Locks</td>
<td>16</td>
</tr>
<tr>
<td>Configuring Form Fields</td>
<td>17</td>
</tr>
<tr>
<td>Troubleshooting Form Scripts</td>
<td>18</td>
</tr>
<tr>
<td>Custom Action to Execute Script</td>
<td>19</td>
</tr>
<tr>
<td>Custom Action to Handle Script Result</td>
<td>19</td>
</tr>
</tbody>
</table>
Background
This guide covers the process of setting up notification templates to include a FormPop URL to provide the recipient of the notification with a link directly to an existing form.

FormPop allows users to view and edit existing Unity Forms using a simplified Web Client viewer interface, without any extra OnBase functionality or access. This can be useful or necessary in a workflow process to collect additional information or obtain signatures or approvals within a form.

Typically, fixed service account credentials are used to allow access to the form, rather than an individual’s OnBase account. This means the form will be available to anyone with the FormPop URL.

This means that beyond the simple setup of the notification and URL, there are important considerations for ensuring modifications can only be made at the appropriate time and by the intended audience.

FormPop does not allow users to create new forms, only to modify existing forms. To allow public submission of Unity forms, please refer to these guides instead:

- Unity Form - Sharing for Public Use (Basic URL)
- Unity Form - Sharing for Public Use (Portal)

NOTE: Only production FormPop URLs can be accessed publicly. For non-production, you’ll need to be on a campus network or VPN to access forms via FormPop. Please keep this in mind when testing.

Prerequisites
To create a FormPop Workflow Notifications, you must have OnBase Configuration and OnBase Studio installed. Refer to the OnBase Client Guides for installation instructions.

Only OnBase Certified Administrators have access to the OnBase Configuration Client. If you need assistance with the Configuration Client steps please contact UIS_DM_SUPPORT@cu.edu.
Resources
For more information, refer to:
- The Unity Form MRG
- The Pop Integrations topic from OnBase Integrations Overview
- The Workflow MRG

If you need assistance, please contact UIS_DM_SUPPORT@cu.edu.

OnBase Configuration Client - Add Document Type to User Group
Add the document type that contains the Unity Form you wish to share via FormPop to the UnityForms Group - X - X group. This is necessary so that adequate permissions are available to view and modify the form without granting any additional access.

NOTE: This does not apply if you are having users authenticate to their own user accounts. In that case, you will still need to make sure the document type is available to the appropriate user groups but it does not need to be assigned to this group used for public submission/modification.

1. Log in to the OnBase Configuration Client

2. Select Users then User Groups/Rights.

3. Find the UnityForms Group - X – X group and select Document Types.
4. Add the appropriate document type to the group by finding it in the list of available document types and moving it to the list of selected document types using the Add button or double clicking.

---

**OnBase Studio - Create the Workflow Notification**

1. Log into OnBase Studio. You will need to be in the Functional - Developer - Workflow group.
2. Click on the **Notifications** tab within the desired repository.

3. Either create a new notification or navigate to the desired notification if it already exists and check it out. Refer to the Workflow MRG for more information if needed.

4. Click in body text of your notification to place the cursor where you want the FormPop URL to be.

5. Click on the **Token** button on the Notification tab of the menu and select **DocPop**.

   Note: In EP3, there is a new “FormPop” option in this menu list, but it will not include all necessary portions of the URL so it will not work! Continue to use the DocPop option and proceed as described below.
6. You will be presented with a URL dialog that you can fill in with the OnBase environment information. Always make sure that the Protocol is changed to https.

7. These are the values that should be used depending on the OnBase environment.

For public form modification (non-OnBase users):

<table>
<thead>
<tr>
<th>SERVER</th>
<th>VIRTUAL_ROOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEV</td>
<td>dm-devunityform.dev.cu.edu UnityForms</td>
</tr>
<tr>
<td>TST</td>
<td>dm-tstunityform.qa.cu.edu UnityForms</td>
</tr>
<tr>
<td>STG</td>
<td>dm-stgunityform.qa.cu.edu UnityForms</td>
</tr>
<tr>
<td>PRD</td>
<td>dm-unityform.prod.cu.edu UnityForms</td>
</tr>
</tbody>
</table>

For users authenticating to their own OnBase accounts (using SSO with campus credentials):
8. The final step is to replace “docpop.aspx” in the resulting URL with “FormPop.aspx”. Simply use your mouse and keyboard to either type it in or copy and paste.

9. This is an example of what the final FormPop notification URL looks like for DMOTST. Make sure not to modify the URL in the notification template.

10. When creating a notification using a FormPop integration, you need to hardcode the server URL for the OnBase environment. Because of this, it’s recommended that you create separate notification templates for each OnBase environment and send the correct one based upon the current environment variable.

   Otherwise, you will need to update or recreate the URL in your notification template(s) after migrating the notification to a different environment.
See the OnBase Guide - Unity Script - How to use the Read Environment Variable Script guide for more information on how to setup and determine the OnBase environment.

If you are using the script, your steps should look like:

<table>
<thead>
<tr>
<th>Description</th>
<th>Option 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action: Run Unity Script</td>
<td>Get Environment Variable</td>
</tr>
<tr>
<td>Description</td>
<td>1</td>
</tr>
<tr>
<td>Action: Set Property Value</td>
<td>Get the environment variable</td>
</tr>
<tr>
<td>Description</td>
<td>Set NotificationName property</td>
</tr>
<tr>
<td>Description</td>
<td>Send template NotificationName</td>
</tr>
</tbody>
</table>

### Action: Run Unity Script
Get the environment variable

- **Action Type**: Run Unity Script
- **General**: Documentation
- **Target**: Current Item
- **Script**: GEN - Read Environment Variable
- **Refresh item after script has executed**: Yes

### Action: Set Property Value
Set a property value to reference the correct notification template

- **Action Type**: Set Property Value
- **General**: Documentation
- **Property Name**: NotificationName
- **Property Value**:
  - **Keyword**: Yes
    - **Target**: Current Item
  - **Work item property**: Yes
  - **Current date/time**: Yes
  - **Current user name**: Yes
  - **Current user email**: Yes
  - **User group name(s) of current user**: Yes
  - **User role name(s) of current user**: Yes
  - **Constant value**:
    - $ - UIS - FormPop - %EnvironmentPropertyName
    - Parse tokens (%K, %D etc.)
    - The value is an array (separated by commas)
    - From E-Form field
**Action: Send Notification**
Send the notification, with the template determined based on the property value

<table>
<thead>
<tr>
<th>Action Type</th>
<th>Send Notification</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Tracking, Documentation</td>
</tr>
<tr>
<td>Attachment</td>
<td>None</td>
</tr>
<tr>
<td>Notification</td>
<td></td>
</tr>
</tbody>
</table>

-OR-

**Description**

1. Option 1
   - Get Environment Variable
   - is environment DMOPRD?
     - On True
     - Send template DMOPRD
     - On False
   - is environment DMOSTG?
     - On True
     - Send template DMOSTG
     - On False
   - is environment DMOTST?
     - On True
     - Send template DMOTST
     - On False
   - is environment DMODEV?
     - On True
     - Send template DMODEV
     - On False

**Action: Run Unity Script**
Get the environment variable

<table>
<thead>
<tr>
<th>Action Type</th>
<th>Run Unity Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Documentation</td>
</tr>
<tr>
<td>Target</td>
<td>Current Item</td>
</tr>
</tbody>
</table>

- OR -

**Script**
- GEN - Read Environment Variable
  - Refresh item after script has executed

---

*University of Colorado*
Boulder | Colorado Springs | Denver | Anschutz Medical Campus*
Rule: Check Property Value
Check if DMOPRD

On True: Send Notification
Send PRD version of template

Rule: Check Property Value
Check if DMOSTG

On True: Send Notification
Send STG version of template
Rule: Check Property Value
Check if DMOTST

On True: Send Notification
Send TST version of template

Rule: Check Property Value
Check if DMODEV

On True: Send Notification
Send DEV version of template
Testing
You will need to thoroughly test the Unity FormPop to make sure it can be processed correctly, the correct portions are visible at each step, the correct portions are editable at each step, etc.

In non-production, you will need to be on a campus network or using a campus VPN to view Unity forms outside the OnBase clients.

Refer to the handbook section on workflow notifications for more information about testing workflow notifications.

You won’t be able to use the Generate Test button on the Notification tab in Studio as the docid and checksum portions of the URL won’t generate unless real content is used to populate them when sending the notification.

Instead, use the Item Generators on the Home tab in Studio to create test content or manually submit forms that will enter the workflow process. Refer to the Studio MRG for more information on using Item Generators.

Securing FormPop Content After Usage
Once a user has the FormPop URL to the form, they may use that link at a later time when you do not want them to access the information or make further changes.

There are a couple of ways you can secure the content from being viewed or accessed later on.

1. Move the document from the original document type to another document type that is not assigned to the UnityForms Group - X – X group.
2. Add Custom Actions to the Unity form to prevent viewing or making modifications to the form after the form has been processed (based on user group, form values, and or workflow queues).

**Custom Action Recommendations to Control Visibility/Editing**

Since every process will have different requirements, you’ll need to determine which of these methods are appropriate or where adjustments are needed. The intention is that these examples will give you some ideas for using custom actions to facilitate the best possible process using formpop. It is important to test all potential scenarios before going live with your solution, as any changes made will only apply to future forms, not existing forms.

**Verifying Identity**

You may wish to verify a user’s identity prior to allowing them to see personal information that’s been filled into the form (in case others are able to access the link). For example, you can require a passcode that is sent to the user or for them to enter their information.

In this case, users are shown a page with only a verification field where they are asked to enter their DOB to proceed.

Once they enter the passcode/DOB and click the verify button, a hidden field's value is set to be used in other custom actions.
Once the criteria are met, users can see page 1 of the form where they need to review the form and provide additional information.

**Form Submission**

Since this method is re-submitting changes to a form that already exists, the user is not routed to a webpage after clicking the Submit button like they are when submitting a form shared to a URL.
However, when the form is submitted, it is reloaded. In order to prevent further changes to the form, you can use a custom action to check for the desired conditions “on load” and show the user a message that the form is no longer available when those conditions are met. For example:

Form is No Longer Available

Once the form is no longer in the queue where external users are being asked to provide information, if the link is used again the user will receive a message that the form is no longer available.

FormPop and Document Locks

Once any document is being viewed or interacted with in OnBase, that document is locked by the user/session doing so. This means that if you are viewing the document in the Unity or web client at the same time you or someone else attempts
to view the document using the formpop URL, the form will be read-only when viewed using the URL (since it is already locked). More information about locks is available in the Unity client MRG.

To help prevent confusion for users who are trying to modify a form that has been locked, you can use these steps to use a script to display a warning message in that situation.

For example, viewing a form through formpop when it’s already locked and therefore read-only:

![Unity script test](image)

vs. in the Unity client (where the lock originated and the form can be edited):

![Unity script test](image)

**Configuring Form Fields**

Certain field IDs must be used for the script to work. For input, you will need:
<table>
<thead>
<tr>
<th>Value/Use</th>
<th>Field ID</th>
<th>Used as</th>
</tr>
</thead>
<tbody>
<tr>
<td>This should be set to the form’s document handle. The easiest way to do this is to add the “Document Handle” property field to the form. The value will then be populated automatically when the form is saved.</td>
<td>formDocumentHandle</td>
<td>Input, required</td>
</tr>
<tr>
<td>You will need to update the field ID to match the expected input for the script.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For outputs, you will need at least one of the following:

<table>
<thead>
<tr>
<th>Value/Use</th>
<th>Field ID</th>
<th>Used as</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date and time for the existing lock</td>
<td>formLockTime</td>
<td>Output, optional</td>
</tr>
<tr>
<td>Data type of field must be Date &amp; Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Username (operID) for the existing lock</td>
<td>formLockUsername</td>
<td>Output, optional</td>
</tr>
<tr>
<td>Real name for the existing lock</td>
<td>formLockRealname</td>
<td>Output, optional</td>
</tr>
<tr>
<td>Will return either True (if a previous lock exists) or False (if no previous lock exists).</td>
<td>formLockResult</td>
<td>Output, optional</td>
</tr>
</tbody>
</table>

**Troubleshooting Form Scripts**

If you’d like to include error messages, you can add the following output fields (these are optional):

<table>
<thead>
<tr>
<th>Value</th>
<th>Field ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Error Message (all scripts)</td>
<td>error_message</td>
</tr>
<tr>
<td>Error Messages for this script only</td>
<td>error_message_GENOnBaseUnityFormLockWarning</td>
</tr>
</tbody>
</table>
Custom Action to Execute Script
A custom action is required to execute a Unity script. Create a custom action with the desired conditions, then add an action to execute the **GEN - OnBase - Unity Form Lock Warning** script.

At a minimum, you should have a condition that the form is not **new** and that the form is **loading**. This script will not work on forms that have not been submitted and saved as documents (since a document handle/ID will not exist) and should be run as the form loads.

Required conditions:
- Form is loading
- Form is not new

You may wish to add other conditions such as:
- User is in the *UnityForms Group - X - X* user group

Required actions:
- Execute script **GEN - OnBase - Unity Form Lock Warning**

Custom Action to Handle Script Result
The script will fill values for any of the output fields that are configured on the form template. You will need to create one or more custom actions to achieve the desired result.

For example, if you want to display a message, it will need to be made visible.
Suggested condition:
  - Expression is true - formLockResult = “True”

You may wish to add other conditions such as:
  - User is in the *UnityForms Group* - X - X user group

Suggested action:
  - Change Visibility - Make paragraphLocked visible

You could also show the information for the user who has the lock on the form if that would be helpful. However, if the public submission account is being used, showing the information for that account may cause more confusion than it prevents, so you may want to include that as a condition in the custom action to only show that information if it indicates a specific user that could be contacted.
Suggested condition:
- Expression is true - formLockResult = “True”
- Expression is true - formLockUsername <> “S_UNITYFORMS_POP”

You may wish to add other conditions such as:
- User is in the UnityForms Group - X - X user group

Suggested action:
- Change Visibility - Make formLockRealname visible

Using a custom action to check for the public account, I see the name of the user with the lock if it is not the public account:
But not when it is the public account: