Operations Orchestration Designer

Software Version: 1.2

Windows and Linux

Use

Document Release Date: September 2017

Software Release Date: September 2017

Hewlett Packard
Enterprise

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notice

Trademark Notices

Support

Visit the HPE Software Support site at: https://softwaresupport.hpe.com/.

Most of the support areas require that you register as an HP Passport user and to sign in. Many also require a support contract. To register for an HP Passport ID, click **Register** on the HPE Support site or click **Create an Account** on the HP Passport login page.

To find more information about access levels, go to: https://softwaresupport.hpe.com/web/softwaresupport/access-levels.

HPE Software Solutions Now accesses the HPSW Solution and Integration Portal website. This site enables you to explore HPE Product Solutions to meet your business needs, includes a full list of Integrations between HPE Products, as well as a listing of ITIL Processes. The URL for this website is https://softwaresupport.hpe.com/km/KM01702731.

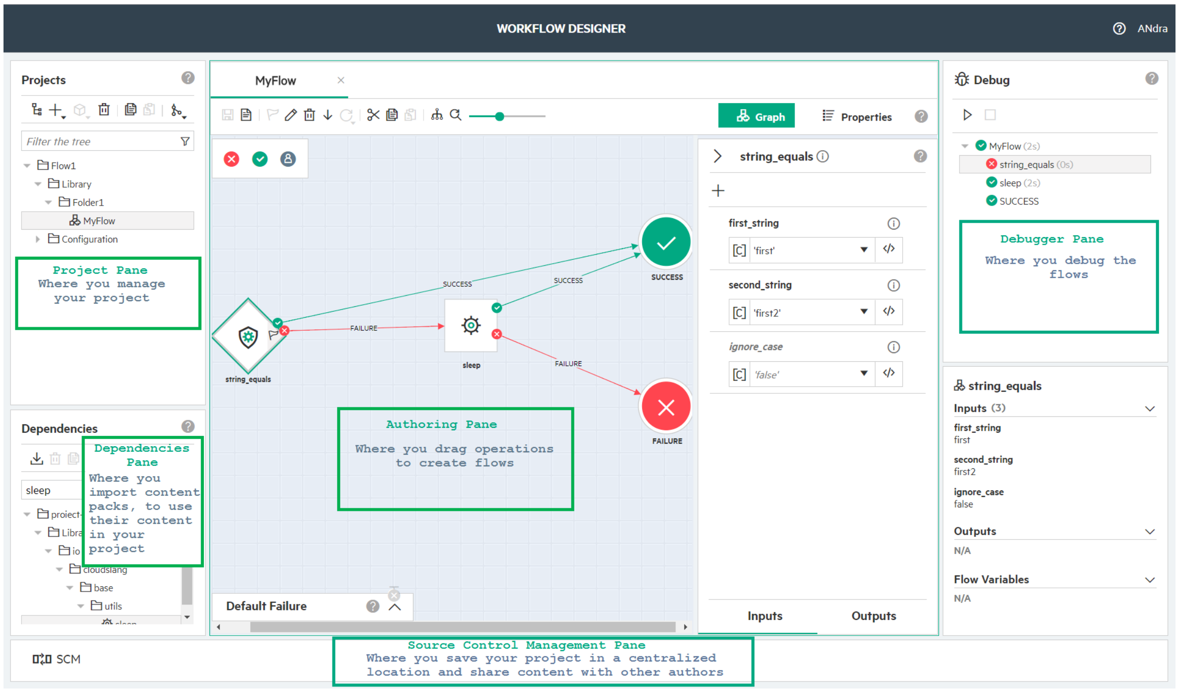
About this PDF Version of Online Help

This document is a PDF version of the online help. This PDF file is provided so you can easily print multiple topics from the help information or read the online help in PDF format. Because this content was originally created to be viewed as online help in a web browser, some topics may not be formatted properly. Some interactive topics may not be present in this PDF version. Those topics can be successfully printed from within the online help.

Contents

Navigate OO Designer

The OO Designer UI includes several sections:



Project Pane
Where you manage your project

Dependencies Pane
Where you import content packs, to use their content in your project

Authoring Pane
Where you drag operations to create flows

Debugger Pane
Where you debug the flows

Source Control Management Pane
Where you save your project in a centralized location and share content with other authors

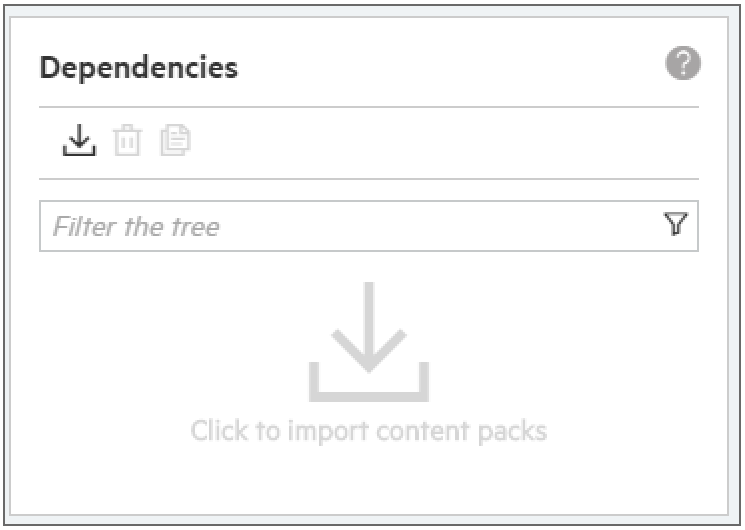
Projects Pane

The **Projects** pane on the top-left shows the project you are currently working on. It displays the editable flows and configuration items that you can use in the project.

For more information about how to work with the OO Designer panes, see "Set up the OO Designer Project" on page 24

Dependencies Pane

The **Dependencies** pane on the bottom-left shows the imported content packs. You can import existing content packs and use its flows, operations and configuration items in your flows.



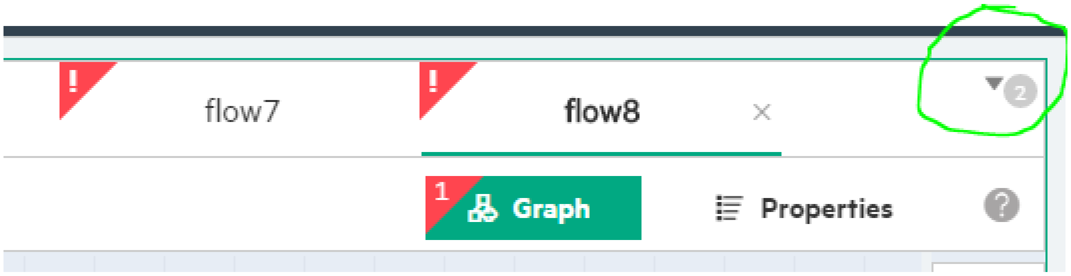
For more information on how to work with the OO Designer panes, see "Set up the OO Designer Project" on page 24.

Authoring Pane

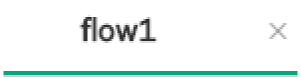
The **Authoring** pane is a rectangular box in the middle. It contains the canvas to work on flows. You can launch a flow onto the canvas area by double-clicking a flow from the **Projects** pane.

If you are working on multiple flows, they are indicated in tabs with the corresponding flow names. Some of these tabs are hidden from view and are displayed in the top-right corner of the authoring pane with a number indicating a number of hidden tabs ▼3.

Click the arrow to view a list of hidden tabs, and then point and click to select a hidden flow to open.



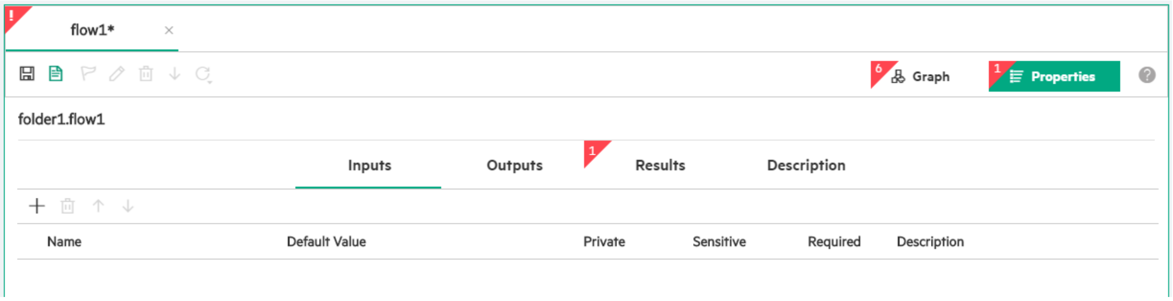
You can close a flow by clicking the **x** displayed next to the flow name.



For more information about how to work with the OO Designer panes, see "Set up the OO Designer Project" on page 24.

Properties Pane

The **Properties** pane contains the **Description**/**Inputs**/**Outputs**/**Results** tabs about the current flow. For details, see "Define Flow Properties" on page 45.

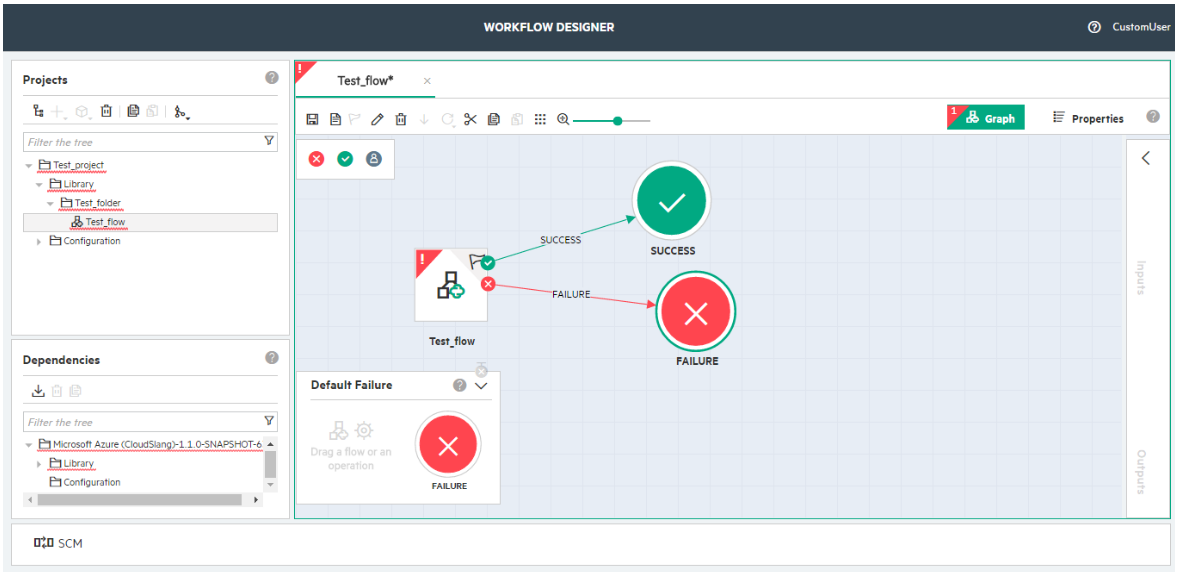


- Click the **Inputs** tab to display the **Inputs** pane. This is where you can add and configure inputs for the flow.
- Click the **Outputs** tab to display the **Outputs** pane. This is where you can add and configure outputs for the flow.
- Click the **Results** tab to display the **Results** pane. This is where you can view the results for the flow, and also configure the descriptions.
- Click the **Description** tab to display the **Description** pane. This is where you can enter the description for the flow.

For more information about how to work with the OO Designer panes, see "Set up the OO Designer Project" on page 24.

Graph Pane

The **Graph** pane is the graphical representation of a flow with the **Inputs** and **Outputs** in tabs.



Select the relevant step on the canvas, and:

- Click the **Inputs** tab at the bottom of the right pane to display the **Inputs** pane. This is where you can view the inputs for the step that correspond to the step operation inputs.
- Click the **Outputs** tab at the bottom of the right pane to display the **Outputs** pane. This is where you can add and configure outputs for the step.

> The selected flows, success, and failures appear with a green outline.

For more information about how to work with the OO Designer panes, see "Set up the OO Designer Project" on page 24.

Debugger Pane

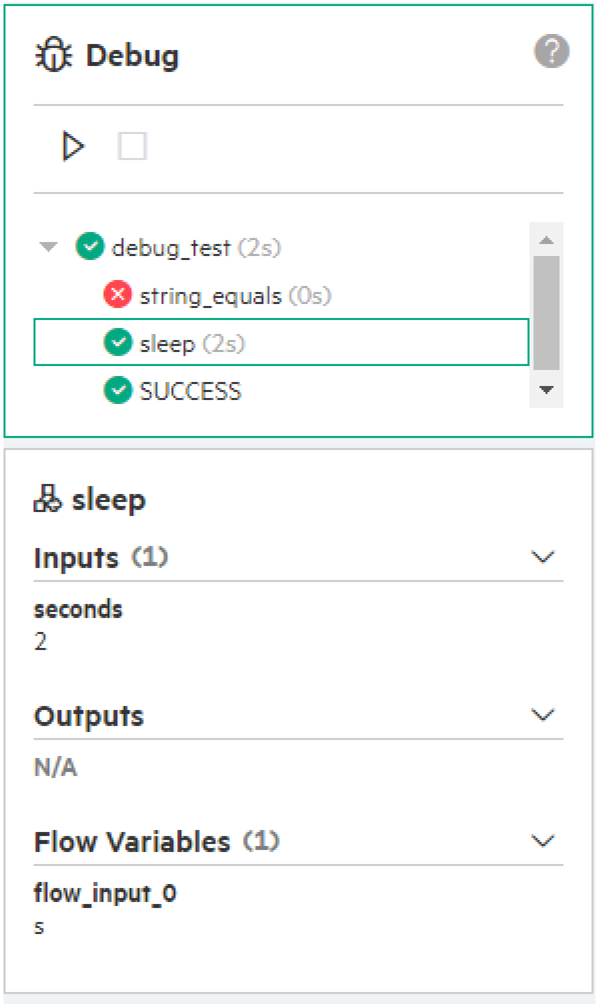
The Debugger pane enables you to test the Flows.

To open the Debugger pane:

1. Select the relevant Flow on the canvas.

> By default, the Debugger pane is in a collapsed state.

2. Click 🐞 or inverted **Debug** text to expand or collapse the pane with the **Run Tree** that has expandable nodes of the step or flow in a graphical mode.
3. Select a node in the tree to display the context of that Flow in the **Context Inspector**



sleep

Inputs (1)

seconds
2

Outputs

N/A

Flow Variables (1)

flow_input_0
5

For more information about how to Debug the Flow, see "Debug the Flow" on page 108

SCM Pane

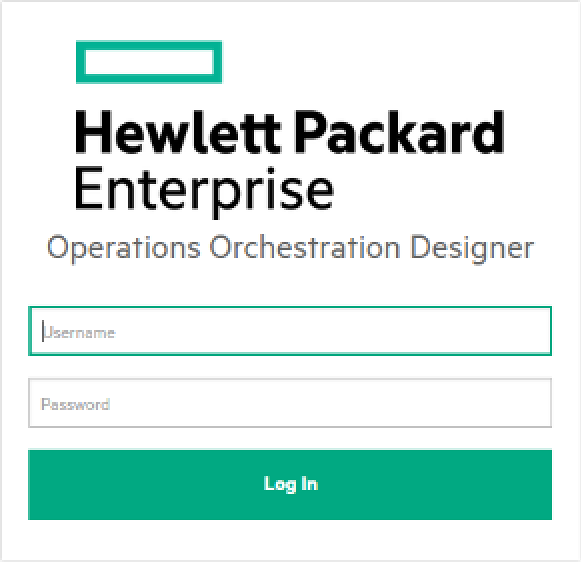
This is where you save your project in a centralized location and share content with other authors. For details, see "Set up and work with Source Control Management" on page 15.

Manage Database User

The IdM Administrator configured during installation of the OO Designer can create new users to access the OO Designer.

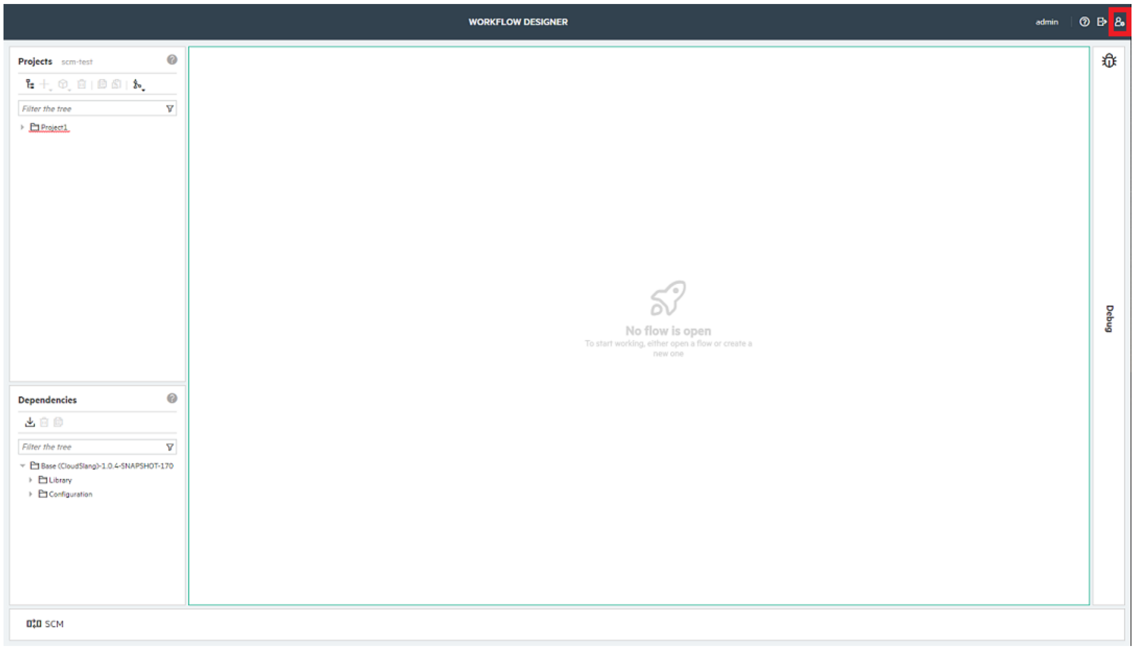
To create new users, complete the following steps:

1. At the Log in page, enter your administrator user name and password, and then click **Login**.



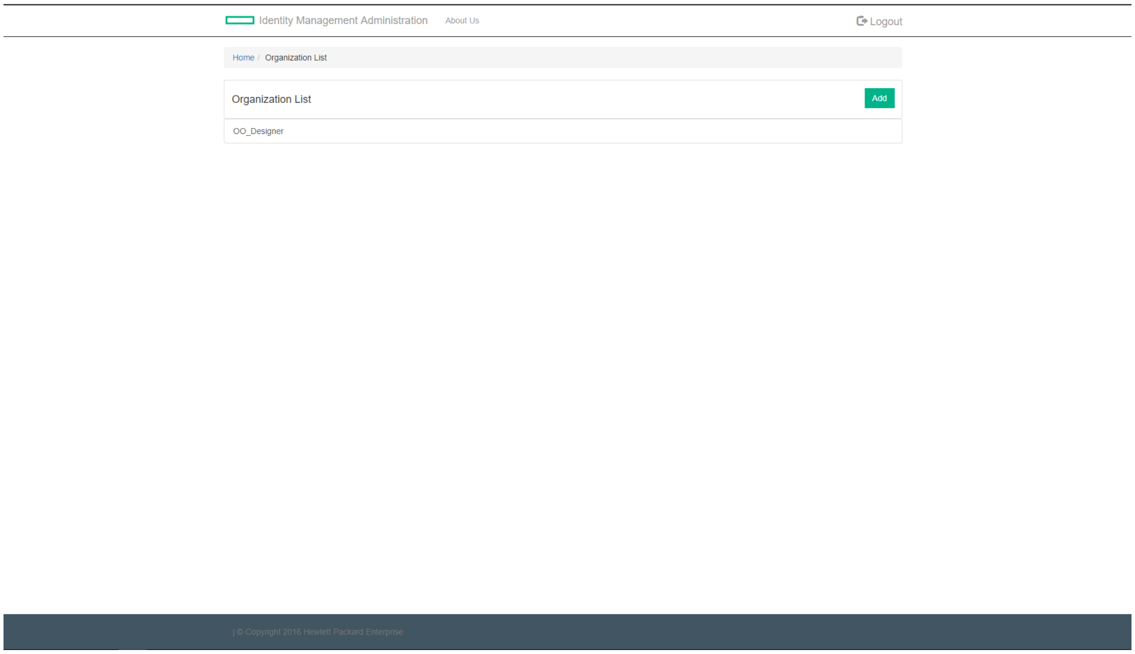
2. In the OO Designer home screen, click the link to IdM Admin as highlighted in the following image.

> The link to IdM Admin icon is visible only to the admin user configured at the time of installation.

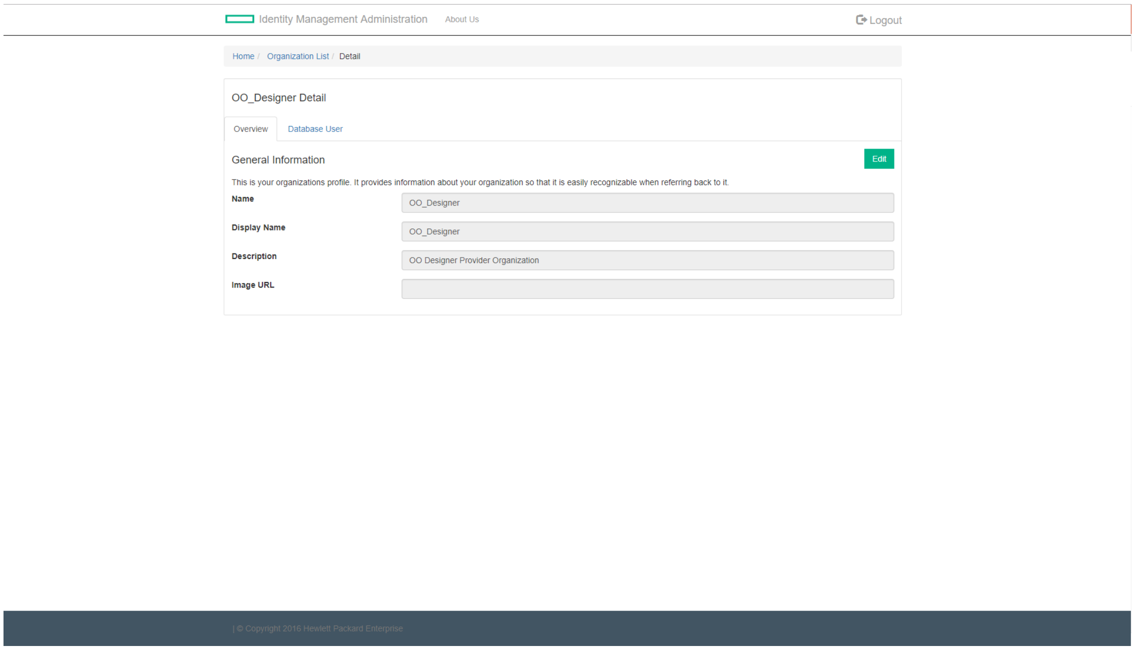


No flow is open
To start working, either open a flow or create a new one

3. Select the **Organization List** tab, and choose your organization.


Identity Management Administration
About Us
Logout
Home / Organization List
Organization List
Add
OO_Designer
| © Copyright 2016 Hewlett Packard Enterprise

4. The general information of the selected organization is displayed as in the following image



Identity Management Administration
About Us
Logout
Home / Organization List / Detail
Overview
Database User
Edit
| © Copyright 2016 Hewlett Packard Enterprise

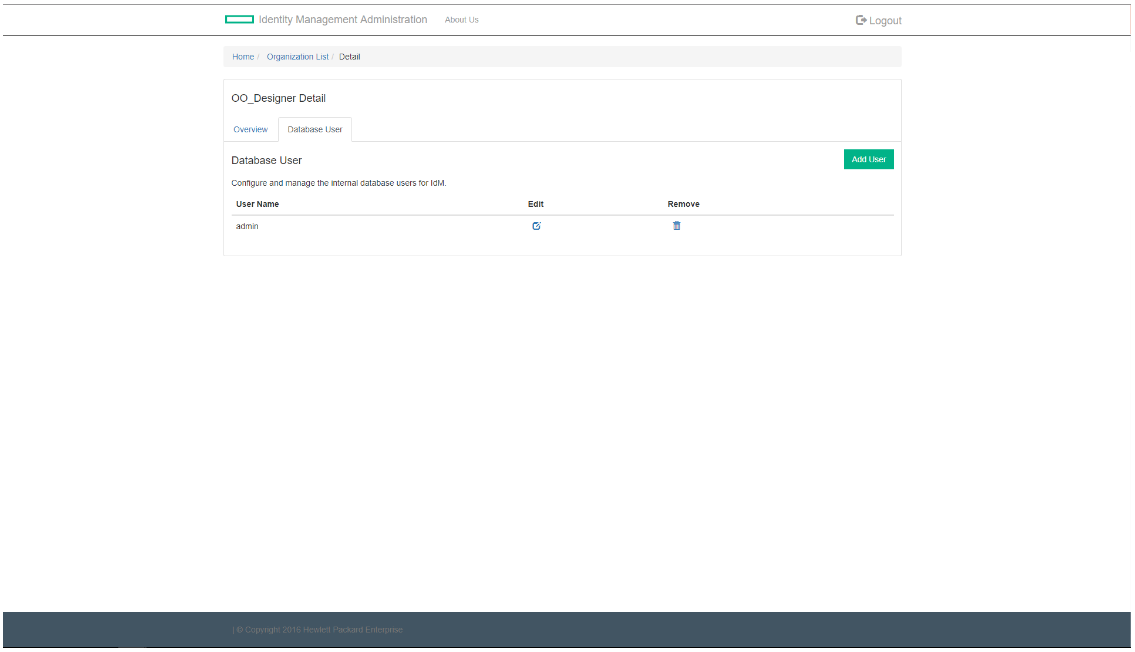
OO_Designer Detail

General Information

This is your organizations profile. It provides information about your organization so that it is easily recognizable when referring back to it.

Name	OO_Designer
Display Name	OO_Designer
Description	OO Designer Provider Organization
Image URL	

5. Select the **Database User** tab, and then click **Add User**.



Identity Management Administration
About Us
Logout
Home / Organization List / Detail
Overview
Database User
Add User
| © Copyright 2016 Hewlett Packard Enterprise

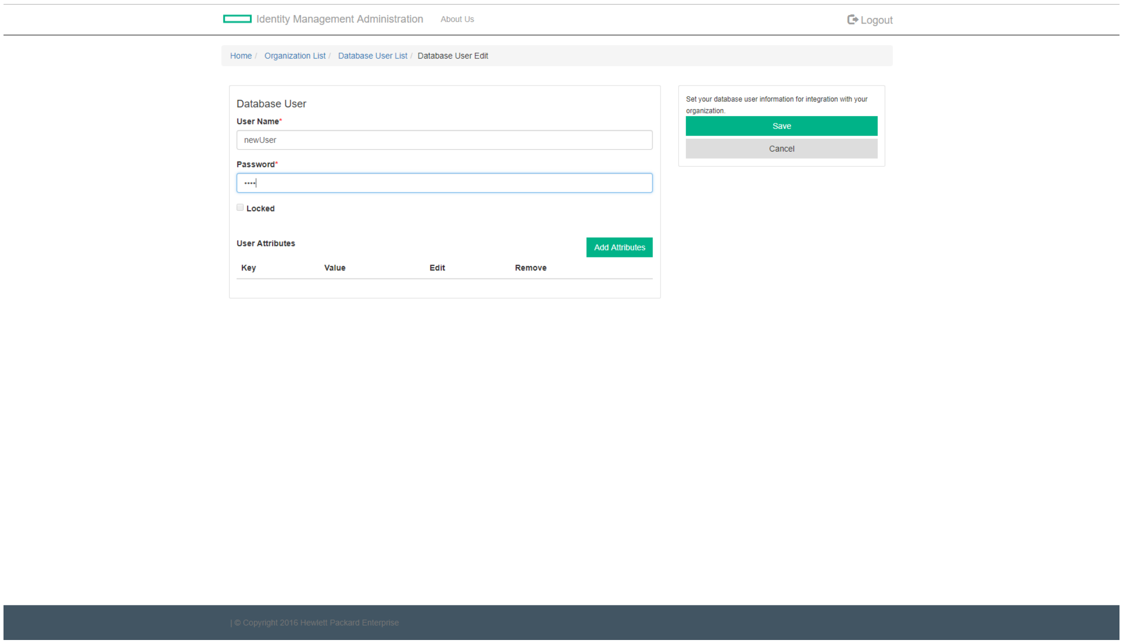
OO_Designer Detail

Database User

Configure and manage the internal database users for IdM.

User Name	Edit	Remove
admin	✎	🗑

6. In the **Database User** section, enter the details for the new user:


Identity Management Administration
About Us
Logout
Home / Organization List / Database User List / Database User Edit
Save
Cancel
Add Attributes
Key
Value
Edit
Remove
| © Copyright 2016 Hewlett Packard Enterprise

Database User

User Name*: newUser

Password*: ••••

□ Locked

User Attributes

Set your database user information for integration with your organization.

a. In the User Name box, enter a unique user name.

b. Enter the password in the Password box.

7. Click **Save** to save the new user.

Set up the Workspace

Before you begin to use the OO Designer, you need to set up your workspace.

To set up your workspace, complete the following tasks:

- Set up **Source Control Management** (SCM) to allow multiple authors to collaborate on the same project. See "Set up Source Control Management for the Project" on the next page.

  > **Note:** If you do not want to use Source Control Management, you can work on a local project. All changes and modifications will be saved only in the database, and you will not be able to recover or revert any of the changes.

- Import **Content Packs** into the workspace to use the content in your projects. See "Import Content Packs to the Dependencies Pane" on page 28.
- Create one or more **Projects**. See "Set up the OO Designer Project" on page 24.
- Set up the **System Properties** for your projects. See "Author System Properties" on page 105

The access to the workspace and the project in OO Designer is user-specific. For example, if authors A and B are using a common computer, and if A logs into the computer, only A's workspace and projects are displayed.

Set up and work with Source Control Management

If you work in a multi-author environment, working with a Source Control Management (SCM) enables you to manage your project in a centralized location and collaborate on it with other authors. It also enables you to recover or revert changes.

Using the Git functionality, you can connect to a remote repository on a server (such as Github, for example) or connect to a shared file system.

Note: Working with Source Control Management is not mandatory, but it is recommended.

To set up a repository, see "Set up Source Control Management for the Project" on the next page.

To work with SCM, see "Work with Source Control Management (SCM)" on page 22.

You can attach only one repository per workspace. Within that repository, it is possible to have multiple projects.

Git Terminology

Pulling Changes

In Git, downloading the latest version of the files from the remote repository is known as a "Pull" action.

After you complete a "Pull" action, new files are added to your projects and files that were modified in Git are updated in your project.

It is recommended to pull the latest version each day, before starting to work on the files.

Committing and Pushing Changes

In some Git applications, there are two steps to syncing your modified files into the repository: "committing" them to a local repository in preparation for upload and "pushing" them to the remote repository.

In OO Designer, there is a single action "commit and push", which you use to sync your files into the remote repository.

Set up Source Control Management for the Project

Using the Git functionality, you can either connect to a remote repository on a server (recommended), such as, *Stash*, or connect to a shared file system.

> **Why do this?** If you work in a multi-author environment, working with a source control management tool enables you to save your project in a centralized location and share content with other authors. It also enables you to recover or revert changes.

In order to set up Source Control Management (SCM) for OO Designer, you need to import a Git repository to your workspace. There are several protocols that you can use to import the repository:

- **HTTPS** (secured)
- **SSH** (secured)
- **File System** (not secured) - use this protocol if you do not have a Git server account

> **Note:** If you already have an unversioned project in your workspace (i.e., saved locally and not

> part of the Git repository), when you import a repository, this project will be added to your local version of the repository. You will be able to commit and push this project to the remote repository. For more information about committing and pushing, see "Work with Source Control Management (SCM)" on page 22.

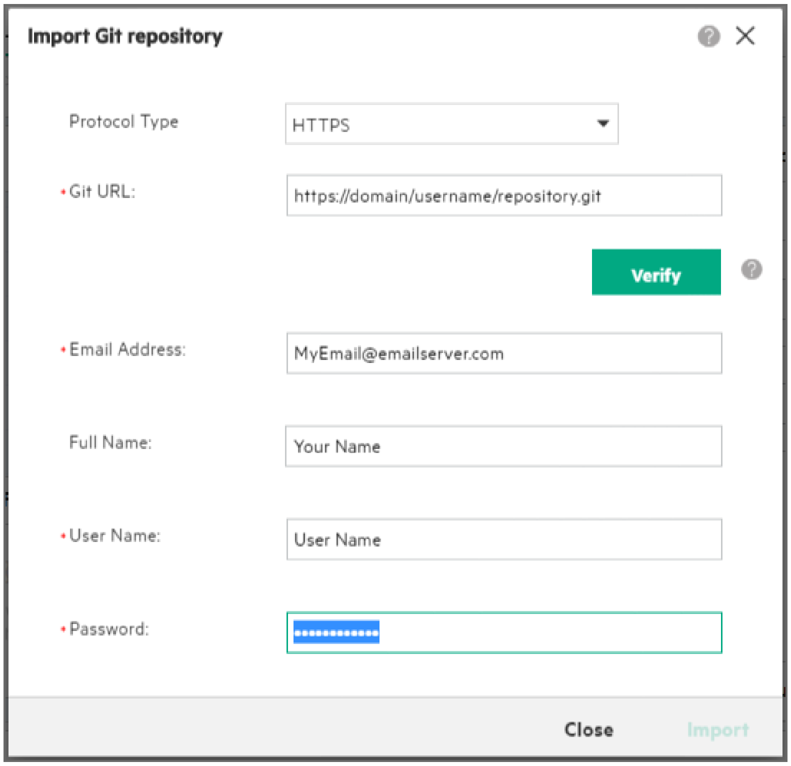
> **Note:** If the projects in the repository do not have the right structure, they will not appear in the project pane and you will not be able to edit them in OO Designer.

Import a Git repository using the HTTPS protocol

1. In the **Projects** pane, click the **Git** button, and then select **Import Repository**.

   The Import Git repository dialog is displayed.

2. Select **HTTPS** as the SCM protocol type.



Import Git repository

Protocol Type: HTTPS

• Git URL: https://domain/username/repository.git

• Email Address: MyEmail@emailserver.com

Full Name: Your Name

• User Name: User Name

• Password: •••••••••••••

3. In the **Git URL** field, enter the URL of the Git repository in the format: **https://<domain>/<username>/<repository_name>.git**.

4. Enter your Email Address and Full Name.

5. Click **Verify** to check that the Git server you are connecting to is trusted.

6. After the verification is complete, enter your user name and password and click **Import**.

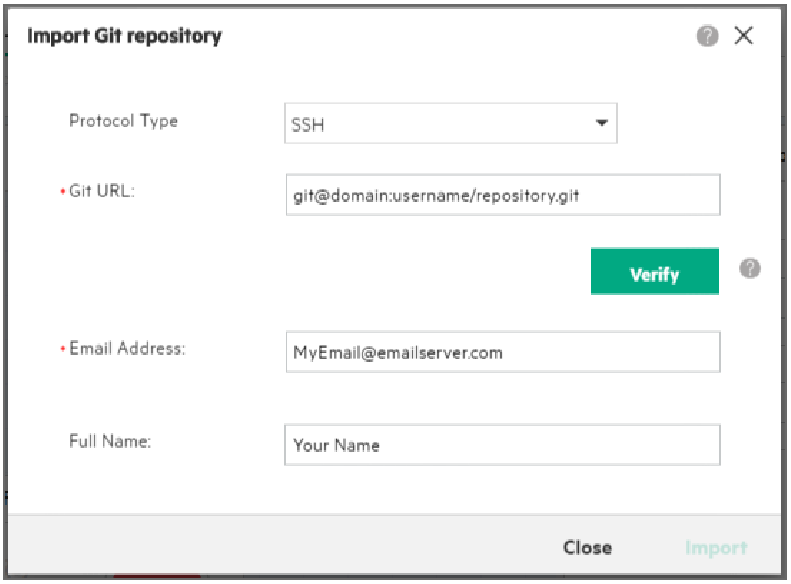
OO Designer verifies that the repository exists in the location, and imports the repository.

7. When the import has been completed successfully. click **Close**.

   The imported repository appears. All the projects in the repository are added to the **Projects** pane.

Import a Git repository using the SSH protocol

1. In the **Projects** pane, click the **Git** button, and then select **Import Repository**.
2. Select **SSH** as the SCM protocol type.



Import Git repository

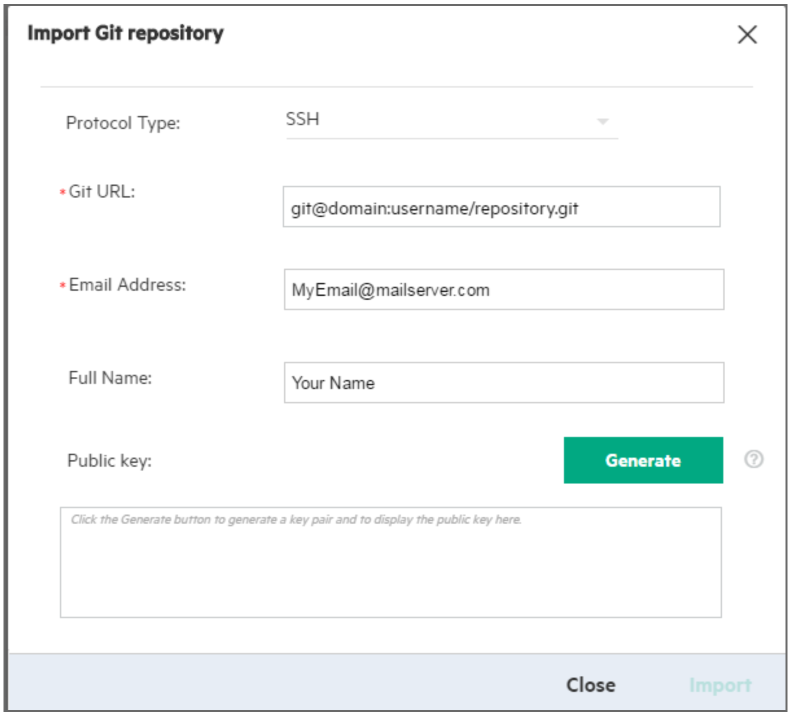
Protocol Type: SSH

* Git URL: git@domain:username/repository.git

* Email Address: MyEmail@emailserver.com

Full Name: Your Name

3. In the **Git URL** field, enter the URL of the Git repository in the format: **git@<domain>:<username>/<repository_name>.git**.
4. Enter your Email Address and Full Name.
5. Click the **Verify** button.
6. After the verification is complete, the **Generate** button appears. Click **Generate** to create an SSH public key.



Protocol Type: SSH

*Git URL: git@domain:username/repository.git

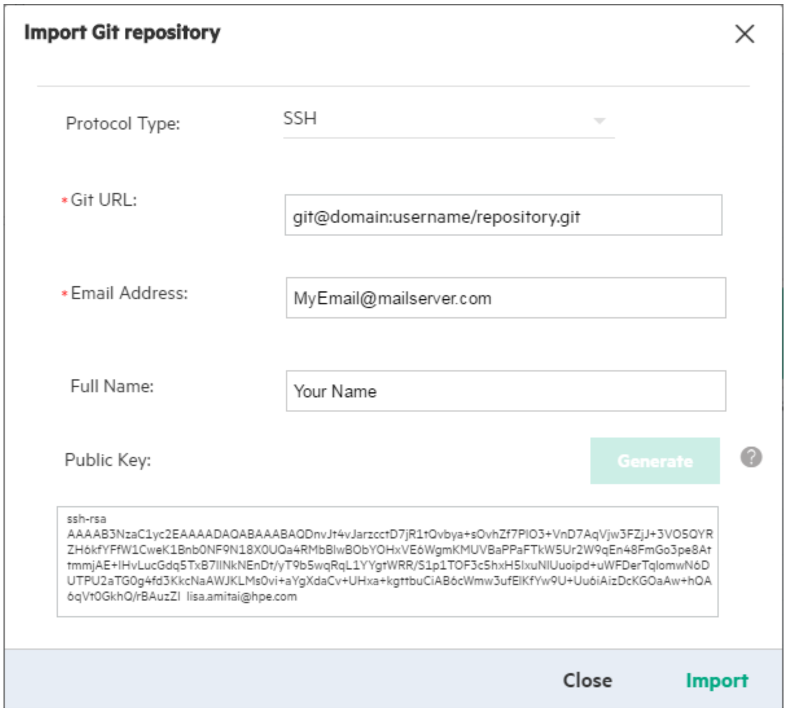
*Email Address: MyEmail@mailserver.com

Full Name: Your Name

Public key:

Click the Generate button to generate a key pair and to display the public key here.

7. After the key has been generated, copy the key. For example:



Protocol Type: SSH

*Git URL: git@domain:username/repository.git

*Email Address: MyEmail@mailserver.com

Full Name: Your Name

Public Key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDnvJt4vJarzcctD7jR1tQvbya+sOvhZf7PIO3+VnD7AqVjw3FZjJ+3VO5QYR
ZH6kfYFfW1CweK1Bnb0NF9N18X0UQa4RMbBlwBObYOHxVE6WgmKMUVBaPPaFTkW5Ur2W9qEn48FmGo3pe8At
tmmjAE+IHvLucGdq5TxB7IlNkNEnDt/yT9b5wqRqL1YYgtWRR/S1p1TOF3c5hxH5IxuNlUuoipd+uWFDerTqlomwN6D
UTPU2aTG0g4fd3KkcNaAWJKLMs0vi+aYgXdaCv+UHxa+kgttbuCiAB6cWmw3ufElKfYw9U+Uu6iAizDcKGOaAw+hQA
6qVt0GkhQ/rBAuzZI  lisa.amitai@hpe.com
```

8. Go to your Git server and copy the SSH public key to your user account.

9. Go back to the **Git Import Repository** dialog box and click **Import**.

   OO Designer verifies that the repository exists in the location, and that authentication is successful via the public\private key pair, and then imports the repository.

10. When the import is completed successfully, click **Close**.

    The imported repository is displayed. All the projects in the repository are added to the **Projects** pane.

Import a Git repository using the File System protocol

Use this protocol, if you do not have a Git server account.

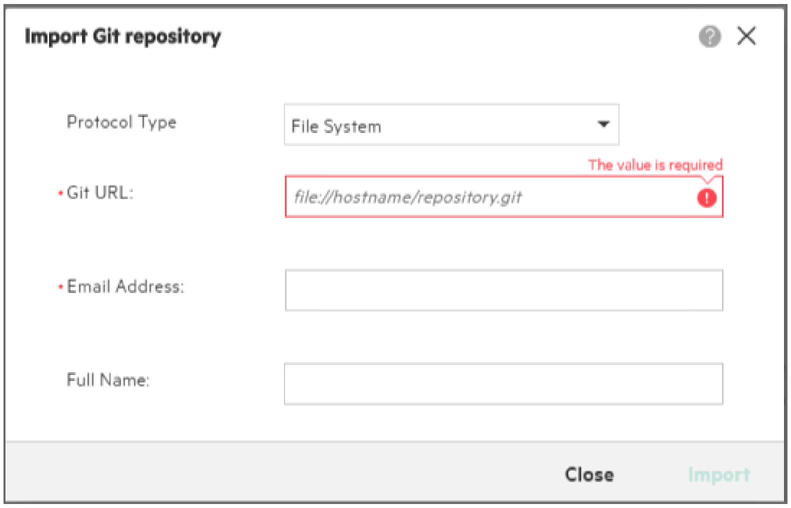
> **Note:** The **File System** protocol is not as secured as **HTTPS** and **SSH** protocols.

If you don't already have a local Git repository, you must create one using the following file:

- **For Windows:** `{OO Designer root folder}/designer/bin/git-create-repo.bat`
- **For Linux:** `{OO Designer root folder}/designer/bin/git-create-repo.bin`

If you already have a local Git repository, start from step 3.

1. Use the **git-create-repo.bat** file to create a local Git repository, as described in "Creating a Git Repository" in the *OO Designer Administration Guide*.
2. Make a note of the URL of the local Git repository.
3. In the **Projects** pane in OO Designer, click the **Git** button and then select **Import Repository**.
4. Select **File System** as the SCM protocol type.



Import Git repository

Protocol Type: File System

The value is required

• Git URL: file://hostname/repository.git

• Email Address:

Full Name:

5. In the **Git URL** field, enter the URL of the Git repository that you created: **file://<hostname or IP>/<shared folder>/<repository_name>.git**

6. Enter your Email Address and Full Name.

7. Click the **Import** button.

   OO Designer verifies that the repository exists in the location, and imports the repository.

8. When the import is completed successfully, click **Close**.

   The imported repository appears. All the projects in the repository are added to the **Projects** pane.

Delete a Git repository from the workspace

1. In the **Projects** pane, click the **Git** button, and then select **Delete Repository**.

2. Click **OK** to confirm the deletion.

   > All the imported projects from your workspace and data not pushed will be removed. The files that you created and were not pushed to the remote repository will be lost.

Edit a Git repository in the workspace

1. In the **Projects** pane, click the **Git** button, and then select **Edit Repository Settings**.

2. The **Edit Git Repository** dialog box fields vary according to the protocol type that was used to import the repository. Before you edit, review the fields that are read-only and editable for three protocols.

   - **File System** protocol - you can edit the full name and email address.
   - **SSH** protocol - you can edit the full name and email address.

     > **Tip:** To generate a new SSH public key, you must re-import the Git repository.

   - **HTTPS** protocol - you can edit the full name, email address, user name, and provide a new password.

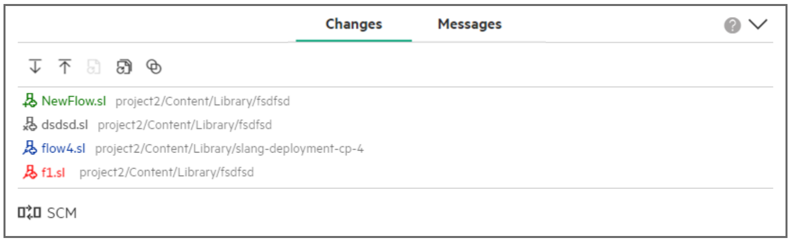
   > **Note:** You cannot edit the repository path. To change this, you need to delete the existing repository and import the new one.

3. Click **Save**.

Work with Source Control Management (SCM)

Track local changes

1. Click the **SCM** ⇄ SCM button to open the **SCM** pane.
2. Click the **Changes** tab to see the changes that were made to the folders and files. Items that were modified, deleted, or added appear in distinct colors.



NewFlow.sl project2/Content/Library/fsdfsd
dsdsd.sl project2/Content/Library/fsdfsd
flow4.sl project2/Content/Library/slang-deployment-cp-4
f1.sl project2/Content/Library/fsdfsd

- **Green** - indicates that a new item has been added to the project, locally.
- **Grey** - indicates that an item has been removed from the project, locally.
- **Blue** - indicates that an item has been edited.
- **Red** - indicates that an item is in conflict.

Pull the most recent changes from the repository

Before starting to work, you must import the most recent changes from the repository to your workspace.

To import, complete the following steps:

1. Click the **SCM** ⇄ SCM button to open the **SCM** pane.
2. Click the **Pull** ↧ button to download the most recent changes to your workspace.

Commit and push your changes to a remote repository

After you make changes locally, you must upload these changes to the remote repository.

To upload, complete the following steps:

> **Note:** In some applications, there are two steps: "committing" files to a local repository in preparation for upload and "pushing" them to the remote repository. In OO Designer, these steps are merged into a single action.

1. Make sure you have saved all your changes.
2. Click the **SCM** SCM button to open the **SCM** pane.
3. Click the **Commit and Push** ↥ button to upload your most recent changes to the remote repository.

   > **Note:** All changes will be committed and pushed to the remote repository.

4. In the **Commit and Push Changes to the Remote Repository** dialog box, enter a description for the changes that were made, and then click **Commit & Push**.

   The changes are saved to the remote repository.

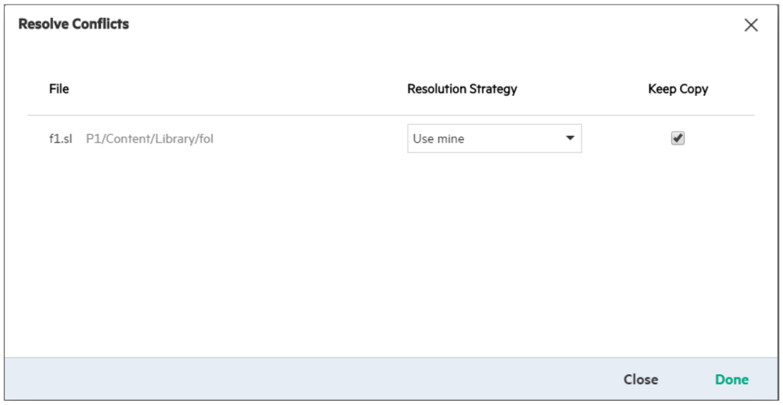
Resolve conflicts

Conflicts may arise if another author has made change to a file you are currently modifying. If the changes occur in different lines in the file, Git is able to automatically merge the two versions. If there are two sets of changes in the same line, they cannot be merged and you need to decide which of the 2 versions you want to keep.

> **Note:** Items that are in conflict appear in red in the **Changes** pane.

When there is a conflict, you can solve it in the **Resolve Conflicts** dialog box, by choosing whether to keep your version of the file (**Use mine**) or the version that it is in conflict with (**Use theirs**).

1. When a conflict occurs (for example, when you try to pull a file from Git and another user has pushed a conflicting version of that file), the **Resolve Conflicts** dialog box appears.



Resolve Conflicts

File	Resolution Strategy	Keep Copy
f1.sl P1/Content/Library/fol	Use mine	☑

2. Under **Resolution Strategy**, select one of the following:
   - **Use mine** - to use your version of the file to the repository
   - **Use theirs** - to keep the file that the other user created

3. Select the **Keep Copy** check box to save a local copy of the version that was not selected under **Resolution Strategy**.
4. Click **Done**.

Revert to a previous version

You can revert changes that you made to an individual file or revert everything to a previous version. This action discards the changes that you have made.

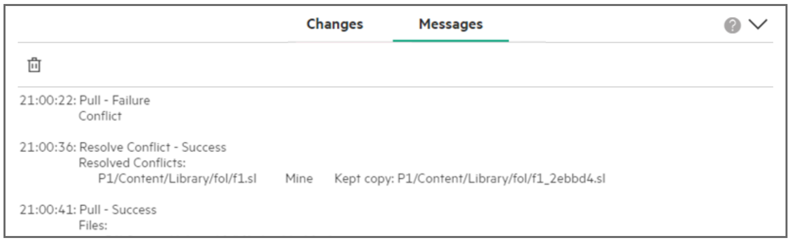
- To revert changes for one file: Select the file in the **SCM** pane and click the **Revert changes for the selected file** button.
- To revert all changes, click the **Revert all changes** button.

  > **Note:** In case you Revert from a state of conflict (conflict dialog was closed without resolving), you will get the version of the file prior to triggering the conflict. You can then also Revert once again to discard your not pushed changes.

View the SCM message history

The **Console** tab is a console that displays a record of all the SCM actions that were taken in this session.

1. Click the **SCM** button to open the **SCM** pane.
2. Click the **Console** tab.



```
21:00:22: Pull - Failure
          Conflict

21:00:36: Resolve Conflict - Success
          Resolved Conflicts:
             P1/Content/Library/fol/f1.sl      Mine     Kept copy: P1/Content/Library/fol/f1_2ebbd4.sl

21:00:41: Pull - Success
          Files:
```

3. To clear the console, click **Delete** .

Set up the OO Designer Project

Before starting to create flows, you need to set up a project in which to work.

Naming Projects and Folders

Names must be unique within their folders.

Names are not case-sensitive, so names such as "Ping" and "ping" are considered duplicates. However, when calling an item, make sure to use the correct case.

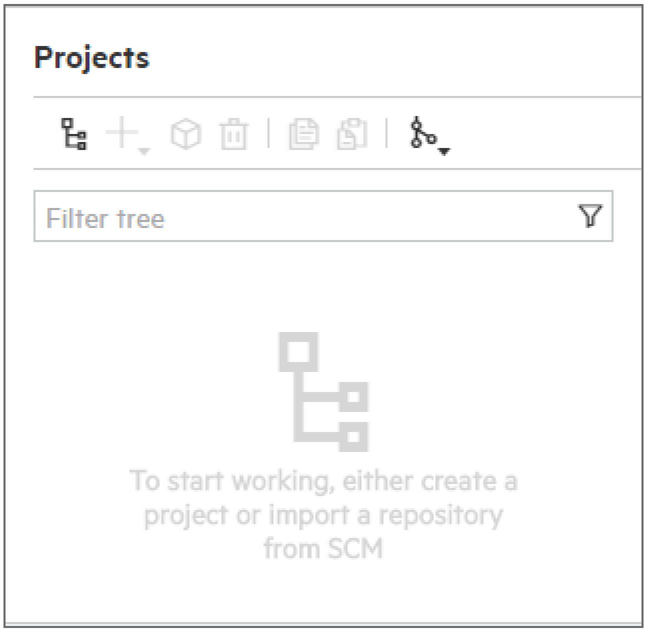
Names can contain alphanumeric characters (A-Z, a-z, 0-9), hyphens (-), and underscores (_).

Project names and Folder names cannot contain white spaces, parentheses (()), square brackets ([]), and curly braces ({}).

Names cannot be identical to the following Windows reserved words: **CON**, **PRN**, **AUX**, **CLOCK$**, **COM1**, **COM2**, **COM3**, **COM4**, **COM5**, **COM6**, **COM7**, **COM8**, **COM9**, **NUL**, **LPT1**, **LPT2**, **LPT3**, **LPT4**, **LPT5**, **LPT6**, **LPT7**, **LPT8**, **LPT9** .

Create a project

1. In the **Projects** pane, click the **Create New Project** button.



To start working, either create a project or import a repository from SCM

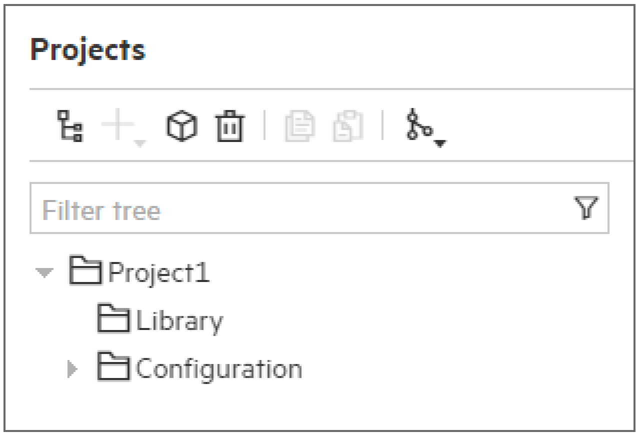
2. Enter a name for the project and click **OK** or press the **Enter** key.

> **Note:** See Naming Projects and Folders.

The new project is displayed in the **Projects** pane.

> **Note:** The new project already includes the basic folder hierarchy: the **Library** folder, where you will store your flows decisions, and operations and the **Configuration**/**System**

> **Properties** folder for the project's system property files.



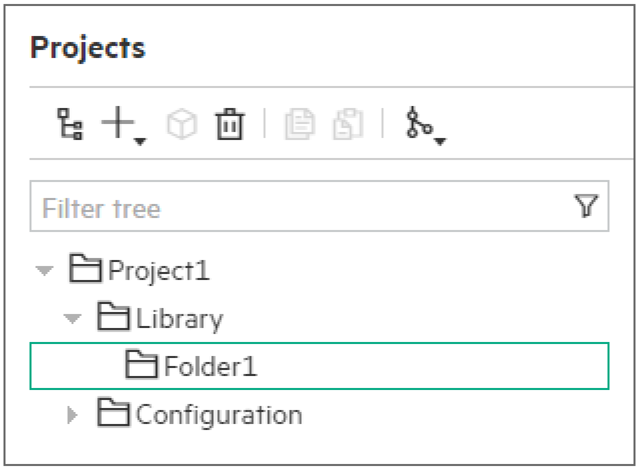
3. To delete an existing project, select the project in the **Projects** pane and click **Delete** 🗑. Click **OK** in the confirmation dialog box.

Manage the folders in the project

Before you can create flows, you must create at least one folder under **Library**.

It is recommended to set up and organize the folders in your project before you start creating flows.

1. To add a folder to the project, select **Library** or an existing folder.
2. Click **New** + and select **Folder**.
3. Enter a name for the folder and click **OK** (or press the **Enter** key).



You can now add subfolders or create flows in the folder.

> **Note:** See Naming Projects and Folders.

4. To delete a folder, select it and click the **Delete** 🗑 button, and then click **OK** to confirm.

    > **Note:** Deleting a folder will delete all of its contents including subfolders and their contents, and will close all open tabs of files that were under that folder.

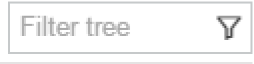
Manage the flows, decisions, and operations in the project

1. To delete flows, decisions, or operations: Select the item, and then click **Delete**. Click **OK** to confirm.

    Deleting a flow will invalidate any other flows that depend on it.

    > **Note:** If you copied an item from a content pack and then deleted the item from the project, this does not affect the content pack that it was copied from.

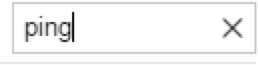
2. To filter the specific items to be displayed in the project pane, enter some text into the **Filter** box in the **Projects** pane.

    For example, enter "ping" to display only flows, decisions, operations, folders, and system properties files containing "ping".

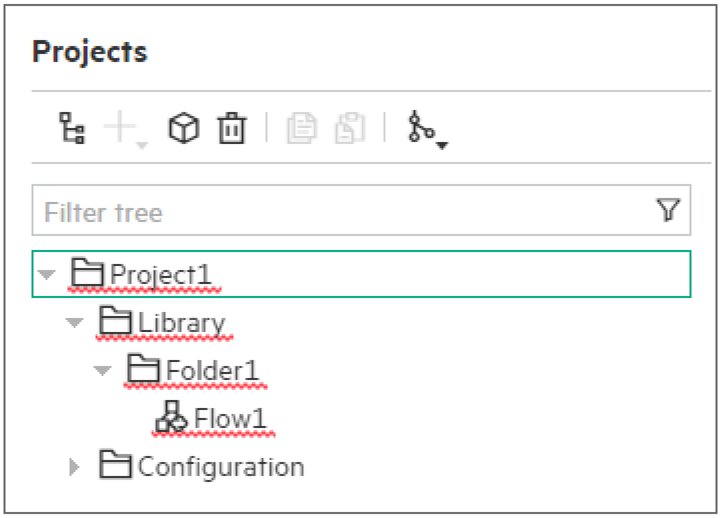
    > **Note:** If the text is part of a folder name, all the contents of that folder, including sub-folders, are displayed.

3. To remove a filter, click **x** at the corner of the **Filter** box.

    
    ping

Check for errors in the project

If there is an error or invalid content anywhere in the **Project** pane (in a flow, operation, decision. or system property), the item with the error is displayed with a red underline. All flows that use this item and the folders above it are also displayed with a red underline.



When you fix the error, the relevant red lines disappear.

For more information about errors and invalid content, see "Check for errors in the content" in "Create and Deploy a Content Pack" on page 112.

Import Content Packs to the Dependencies Pane

You must import each Content Pack (CP) that you want to use even if it was imported by another author connected to the same OO Designer.

A content pack is a collection of operations, flows, actions (Java-based or .Net based), decisions, configuration items (such as selection lists, domain terms, and so on) and resource bundles.

You can import content packs containing content that was authored using CloudSlang.

See Create a content pack with CloudSlang content for how to prepare the content packs to be imported into OO Designer.

After importing a content pack

Once you have imported a content pack, these items are available in the **Dependencies** pane as read-only, and you can use them in your flows. For details, see "Dependencies Pane" on page 6.

> **Important!** If you modify and then import a content pack that was already imported to OO Designer (for example, by a different user), it ***must*** have an updated version number in order for the changes to take effect. If you import a content pack with the same name and version number as one that was already imported, the new content pack will reference the original one and ***will not replace it***.

After you have imported the desired content packs into the workspace, you can copy flows and system properties into the project (by copying from the **Dependencies** pane and pasting into the **Projects** pane).

Once flows are copied into the project, you can edit them in the OO Designer Authoring pane.

> **Why do this?** When you create a step from an operation/flow, the step is an instance of that operation/flow.
>
> Items in the **Dependencies** pane are read-only. If you create a step from a flow from there, you will be able to modify that step only. If you want to make changes to a flow, in order to use the modified version for multiple steps, you need to copy it from the **Dependencies** pane and paste it into the **Projects** pane.

How to use CloudSlang files created outside OO Designer in OO Designer

See "How to use CloudSlang files created outside OO Designer in OO Designer" on page 116.

Import content packs to the workspace

1. Locate your required Content Packs:
   - Use the content packs that were included when you installed OO Designer.
   - Download additional content packs from ITOM Marketplace.
   - Create content packs from existing CloudSlang content, as described in Create a content pack with CloudSlang content.

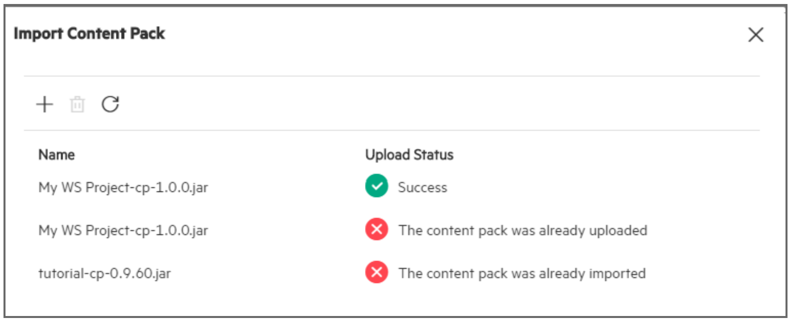
   > **Important!** Make sure that before importing a content pack, all content packs it depends on are already imported. Specifically, import the Base content pack first, because other content packs depend on it.

2. Click the **Import Content Pack** ⤓ button in the **Dependencies** pane.
3. In the Import Content Pack dialog box, click **New** +.
4. In the browse dialog, select one or more content packs and click **Open**.

   > **Tip:** You can also drag and drop from the browse dialog.

5. If all selected content packs are successfully uploaded, the **Import** button is enabled.

   If any of the content packs were not uploaded successfully, the **Import** button is not disabled.



Import Content Pack

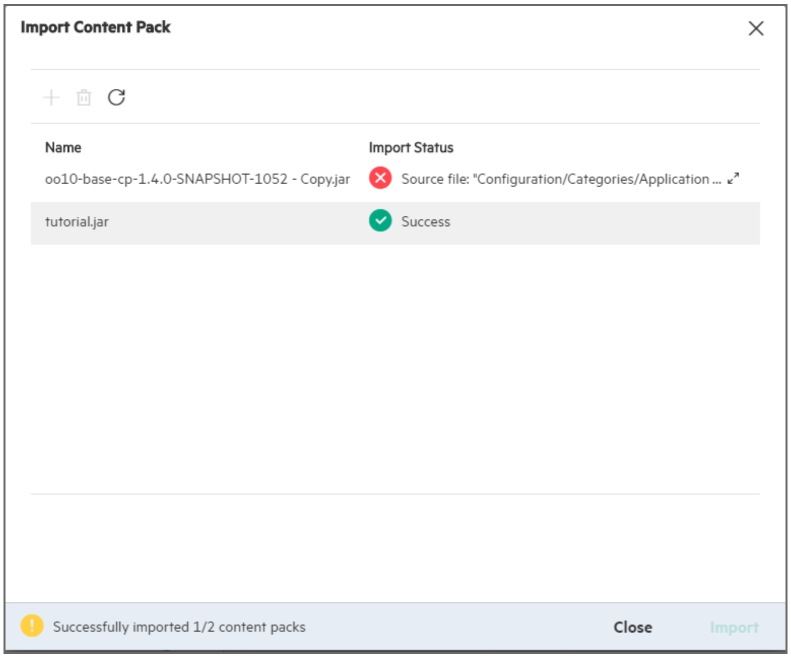
Name	Upload Status
My WS Project-cp-1.0.0.jar	✓ Success
My WS Project-cp-1.0.0.jar	✗ The content pack was already uploaded
tutorial-cp-0.9.60.jar	✗ The content pack was already imported

In this situation, you will need to remove the failed ones from the dialog box. Do one of the following:

- Select a content pack that was not uploaded successfully and click the **Delete** 🗑 button to remove it.
- Click the **Clear** ↻ button to clear all displayed content packs from the dialog box and restart the process of uploading content packs.

> **Note:** The default size limit of content packs that can be uploaded to OO Designer is 200 MB. However, you can configure this size limit. See "Configuring Content Pack Import Size" in the *OO Designer Administration Guide*.

6. After the content packs are successfully uploaded, click the **Import** button.



Import Content Pack

Name	Import Status
oo10-base-cp-1.4.0-SNAPSHOT-1052 - Copy.jar	✗ Source file: "Configuration/Categories/Application ...
tutorial.jar	✓ Success

! Successfully imported 1/2 content packs

7. Click **Close** to close the Import Content Pack dialog box. The successfully imported content packs are displayed in the **Dependencies** pane.

Filter the dependencies

1. To filter the specific items to be displayed in the dependencies pane, enter some text into the **Filter** box in the **Dependencies** pane.

   For example, enter "ping" to display only flows, decisions, operations, folders, and system properties files containing "ping".

   > **Note:** If the text is part of a folder name, all the contents of that folder, including sub-folders, are displayed.

2. To remove a filter, click the x at the side of the **Filter** box.

   ping

View the details of a read-only item from the Dependencies pane

While you cannot edit a flow/operation from the **Dependencies** pane, you can view its details in the Authoring pane.

1. Select the flow/operation in the **Dependencies** pane.

2. Double-click the flow/operation to display it as read-only in the Authoring pane. A **Read-only** icon appears in the tab, next to the name.

Author in OO Designer

Authoring is the process of creating and editing content.

In OO Designer, you can author:

- Operations - See "Author Operations" on page 98
- Flows - See "Author Flows" on page 33
- System properties - See "Author System Properties" on page 105

Author Flows

The major steps in creating a flow are:

1. Create a new flow in the project. See "Create or Edit a Flow" below.
2. Add steps to the flow. Each step is an instance of an operation or another flow. See "Create Steps in a Flow" on the next page.
3. Create navigation lines to connect between steps. See "Create Navigation Lines between Steps in a Flow" on page 38.
4. Add result steps (success, failure, custom) to the end points of the flow. See "Add Result Steps to a Flow" on page 40.
5. Set up inputs and outputs for the flow. See "Define Flow Properties" on page 45.
6. Set up the values of inputs and outputs for steps. See "Define Step Properties" on page 55.
7. Change the start step. See "Change the Start Step" on page 79.
8. Create flows with loops. See "Create Flows with Loops" on page 80.

Create or Edit a Flow

A flow is a composition of steps that forms a set of actions that are linked by decision-making logic in order to automate tasks. For example: Health checks, troubleshooting, remediation, or any other repetitive IT tasks.

Creating flows is the main function of OO Designer.

You can also display and edit a flow that was created externally from OO Designer, for example in ATOM. However, if the flow contains errors, OO Designer may not be able to display it in the Authoring pane. In such a case, you will need to fix the flow in the editor in which it was created and import it to OO Designer again.

Naming Flows

The name of a flow is limited to 128 characters.

The combined name and path of a flow (under the **Library** folder) is limited to 220 characters.

Names must be unique within their folders.

Names are not case-sensitive, so names such as "Ping" and "ping" are considered duplicates. However, when calling an item, make sure to use the correct case.

Flow names must contain alphanumeric characters (A-Z, a-z, 0-9), hyphens (-), and underscores (_).

Flow names cannot contain white spaces, parentheses (()), square brackets ([]), curly braces ({}), and only digits (0-9).

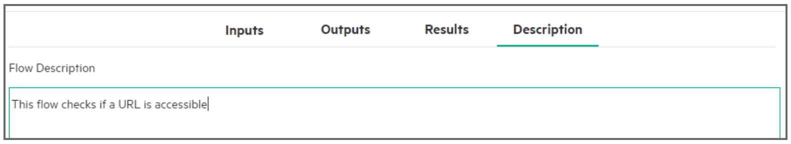
Names cannot be identical to the following Windows reserved words: **CON**, **PRN**, **AUX**, **CLOCK$**, **COM1**, **COM2**, **COM3**, **COM4**, **COM5**, **COM6**, **COM7**, **COM8**, **COM9**, **NUL**, **LPT1**, **LPT2**, **LPT3**, **LPT4**, **LPT5**, **LPT6**, **LPT7**, **LPT8**, **LPT9** .

Create a new flow

1. In the **Projects** pane, select the folder in which you want to locate the new flow.
2. Click the **New** + ▾ button and select **Flow**.
3. In the New Flow dialog box, enter a name for the flow and click **OK**.

   > **Note:** When naming a flow, make sure the naming conventions are followed. See Naming Flows. To change a flow name, copy it as a new flow with the correct name.

   A new tab opens in the **Authoring** pane with the name of the new flow.
4. Click the **Properties** button ☰ Properties to display the flow's **Properties** pane.
5. Click the **Description** tab. You can enter a summary of what the flow does.



Flow Description

This flow checks if a URL is accessible

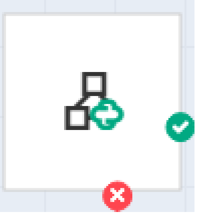
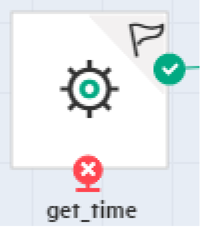
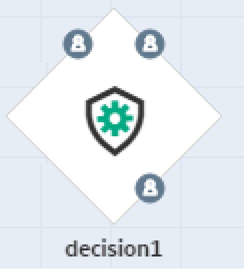
Create Steps in a Flow

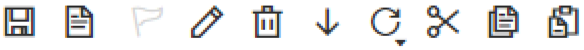
When you create a step from an operation, decision, or flow, the step is an instance of that operation or flow and so inherits the inputs, outputs , and other characteristics.

Create a step from a flow, operation, or decision

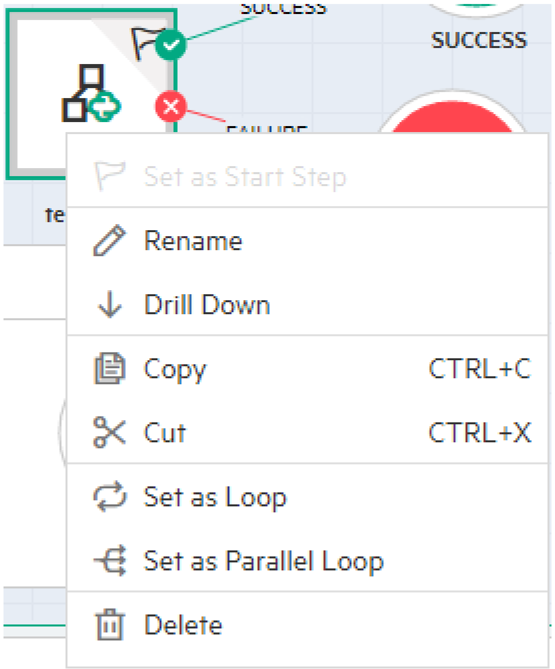
1. In the **Projects** pane, double-click the flow in which you want to create the step, so that it opens in the authoring pane.

2. Drag a flow/operation/decision to the authoring pane, from either the **Projects** pane or the **Dependencies** pane.

> **Note:** The step icon reflects whether the step was created from an operation, a flow, or a decision operation.
>
> | Flow | Operation | Decision |
> |---|---|---|
> |  |  |  |

3. To copy, cut, paste, and rename a step, select it and use appropriate buttons from the Authoring pane toolbar  .

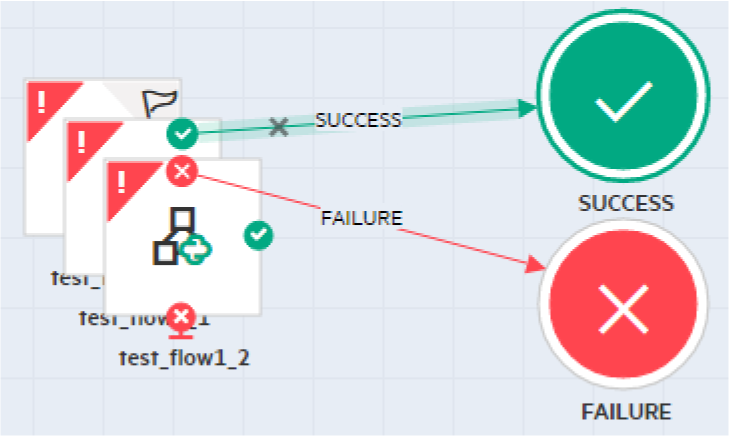
You can also right-click the step and choose the following actions: Set as Start Step, Rename, Drill Down, Copy, Cut, Set as Loop, Set as Parallel Loop, and Delete.



a. To copy a step, click the **Copy** button in the authoring pane toolbar or press CTRL+C keys, then click **Paste** button or press CTRL+V keys to paste.

b. To cut a step, click the **Cut** button in the authoring pane toolbar or press CTRL+X, then click **Paste** button or press CTRL+V keys to paste.

> **Note:**
>
> You can copy and paste steps on the canvas multiple times as per your requirement.



c. To rename a step, click the **Edit** button in the authoring pane toolbar. Enter a name for this step and click **OK**.

For example: you may want to rename a step to reflect its function within the flow.

> **Note:**
>
> - A name of a renamed step must be unique in the flow. This applies to names with different case—for example, you cannot name one step to "Ping" and another to "ping".
> - A renamed step cannot have an blank name.

4. To delete a step from the canvas, select it and click the **Delete** button in the authoring pane toolbar.

> **Note:** When you delete a step, any navigation lines linking this step to other steps are also deleted.

5. To reorganize the steps and results on the canvas, click the auto arrange icon from the authoring pane toolbar.

6. To zoom-in or zoom-out of the canvas, you can use mouse-click and drag or scroll on the slider ━━●──── . To reset the zoom, click 🔍.

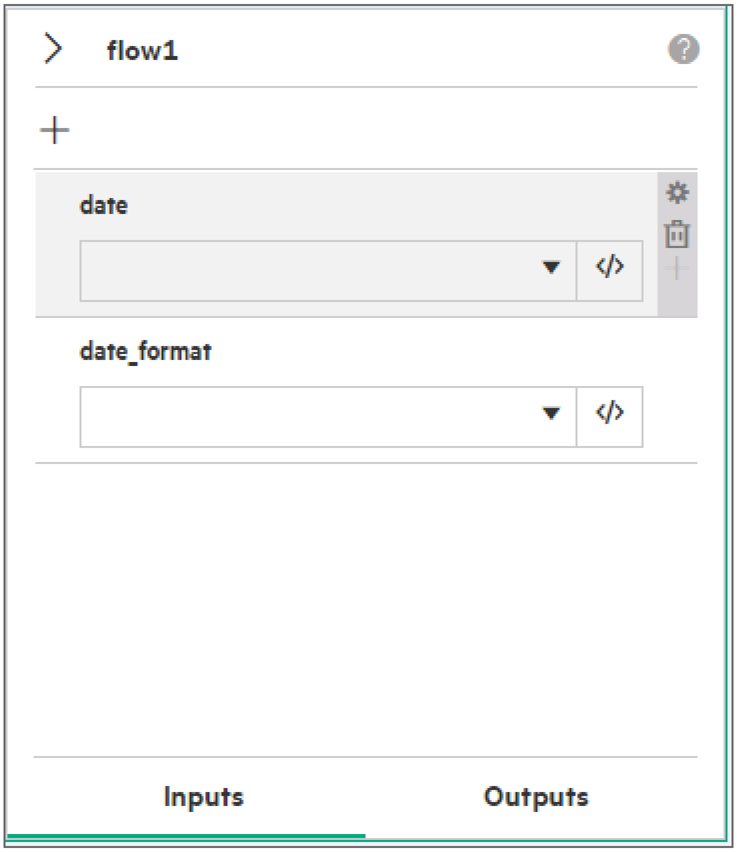
> **Note:** The auto arrange, zoom and reset zoom icons are active and enabled in graph view only.

7. Click **Save** 💾.

View the step properties

1. Click a step in the authoring pane. The **Step Properties** pane appears at the right side of the canvas.

   The **Properties** pane contains the **Input**/**Output** tabs, with information about the selected step.

   For more information about the **Inputs**/**Outputs** tabs, see "Define Step Properties" on page 55.

Drill down to view a subflow/operation within a step

To drill down within a step to see the details of the flow, decision, or operation that was used to create it:

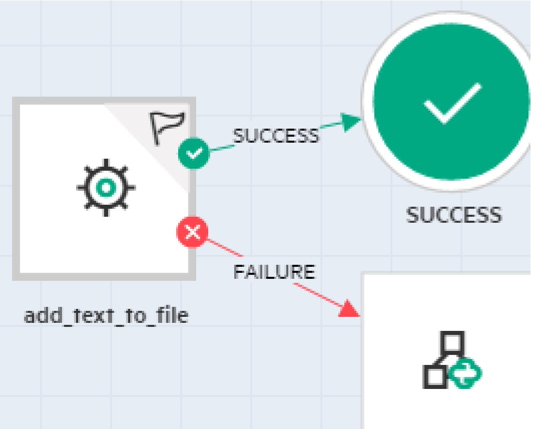
- Click the **Drill Down into the Step**↓ button in the authoring pane toolbar.

Create Navigation Lines between Steps in a Flow

You must connect any two steps in a flow with a navigation line.

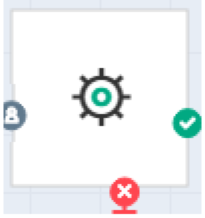
About navigation lines

A navigation line starts from one of a step’s response ports and connects to another step or to a result step.



- Every **Success** or **Custom** response in a flow must have a navigation line either to a subsequent step or to a result step.
- It is optional to connect **Failure** responses to a subsequent step or result step. For more information about result steps, see "Add Result Steps to a Flow" on page 40.

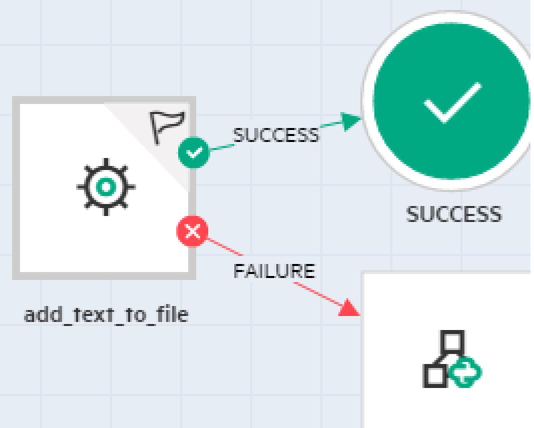
The response ports that appear in a step are in accordance with the responses that were defined for the operation or flow that the step instantiates. For example, if a step instantiates a subflow with three responses: success, failure, and a custom response, the step will have three corresponding response ports.



If a flow already has a navigation line from a step's response port, and that response is deleted from the flow or operation instantiated in the step, in the flow the response port is grayed-out and navigation line appears as a dotted gray line.

Create and Delete the navigation lines between steps

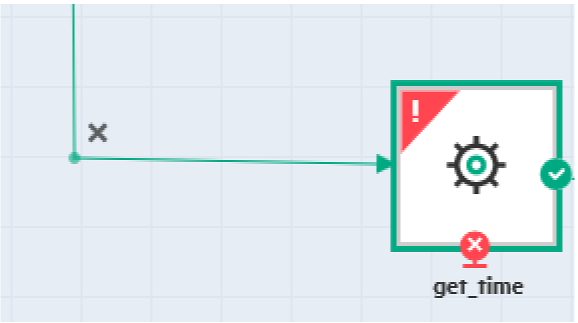
1. On each step that you want to connect to the next step (or to a result step), click the icon that represents one of the response ports, and drag a line to the destination step for that response.



2. Repeat the process for the other response icons on the step.

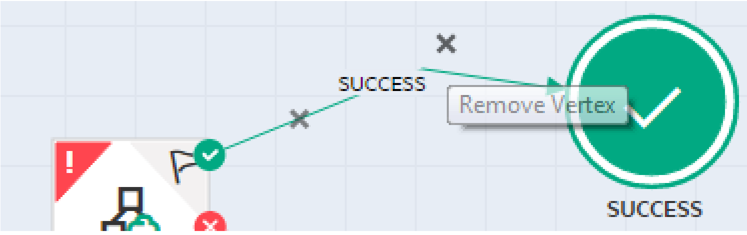
3. To add, move, and remove a vertex from the navigation line, follow the below steps:

   a. To add or move a vertex on the navigation line, click the navigation line and drag it to the desired position.
   A small vertex with **x** icon appears on the navigation line as shown below.

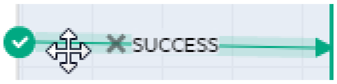


   b. To delete a vertex from the navigation line, hover over the navigation line with the vertex and click **x** icon.

The vertex is removed and the original navigation line without a vertex is displayed.

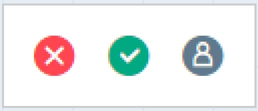


4. To delete a navigation line, hover over the navigation line and click the **x** that appears on the line.



Add Result Steps to a Flow

Result steps return an outcome for the entire flow and end the flow. To create a result step, drag it from the **Results** toolbar.



All **Success** and **Custom** response ports must be connected either to another step or to a result step.

Options for Failure response ports

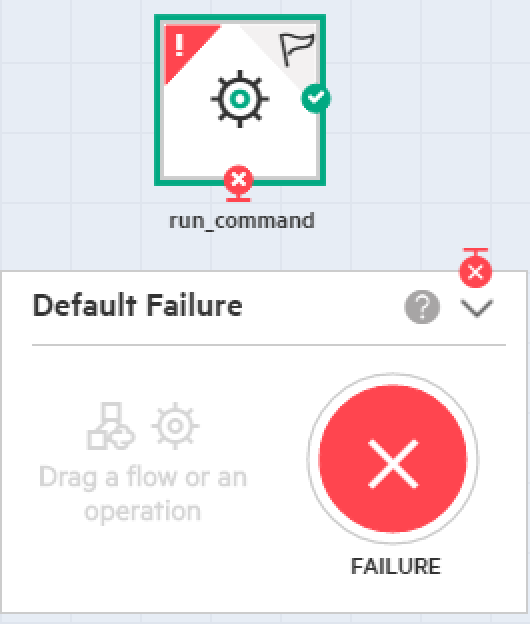
There are two options for **Failure** response ports:

- If you don't connect a step’s **Failure** response port to a step, the response icon will remain with grounding ✖ and will end the flow with a default failure step.

  The default failure step can be configured to run a specific step/flow before failing.

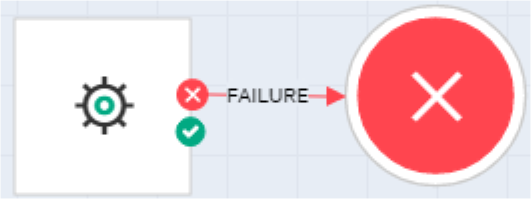
  Use this option if there are a lot of fail points in a flow and you want the same action to occur on each failure.

  You can also use this option if you want the flow to be readable and clean, as it removes the failures from the graph and promotes the successful path of the flow.

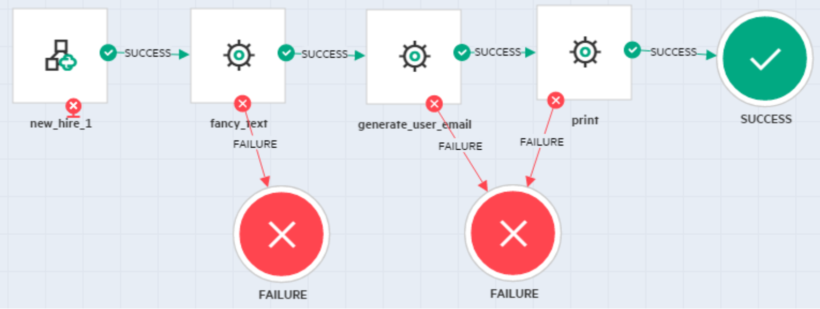


- If you connect a step's **Failure** response port to any step, the grounding disappears, the flow depends on what it is connected to, and does not run the default failure step.

  Use this option if you want to configure the action for a specific fail point separately.

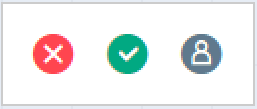


Example of multiple option of how to handle failure in a flow.

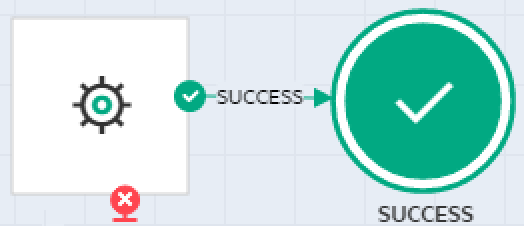


Add Results to the flow

1. To add **Success** and **Custom** result points to the flow, drag result steps from the **Results** toolbar, and place them next to the step connected to it.



2. Click the icon that represents one of the response ports, and drag a line to the result step for that response.

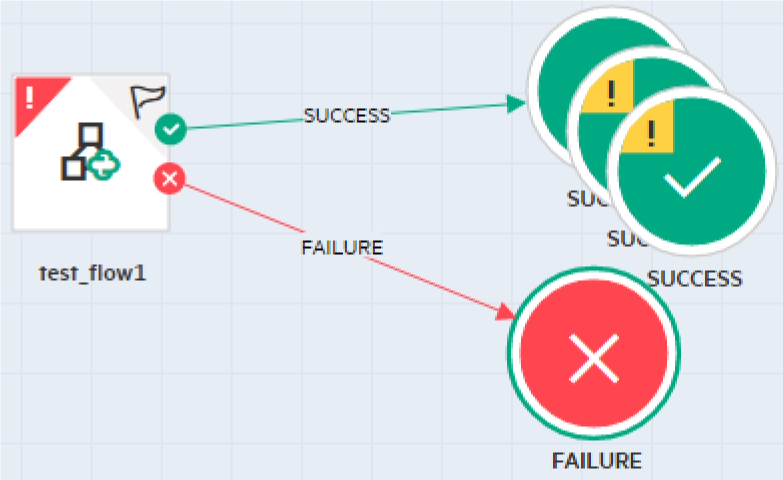


3. To copy, cut, paste, and rename a result step, select it and use appropriate buttons from the authoring pane toolbar .

   a. To copy a result step, click the **Copy** button in the authoring pane toolbar or press CTRL+C keys, then Click or press CTRL+V keys to paste.

   b. To cut a result step, click the **Cut** button in the authoring pane toolbar or press CTRL+X, then Click or press CTRL+V keys to paste.

   > **Note:**
   >
   > - You can copy and paste result steps on the canvas multiple times as per your requirement.



4. To rename a result step, select it and click the **Edit** ✎ button in the authoring pane toolbar.

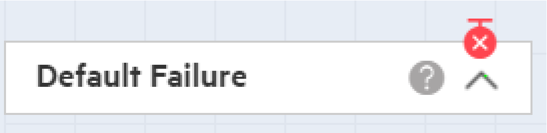
> **Note:**
>
> - If you rename a **Success/Failure** step, the new name must start with the prefix **Success/Failure**.
> - A **Custom** step can have any name.
> - A renamed step must be unique in the flow. This applies to names with different case—for example, you cannot name one step to "Success_All" and another to "success_all".
> - A renamed step cannot have a blank name.
> - You cannot change the type of a response step by renaming.
> - A **Failure** step cannot be named "on_failure".

(Optional) Set up a default failure step for the flow

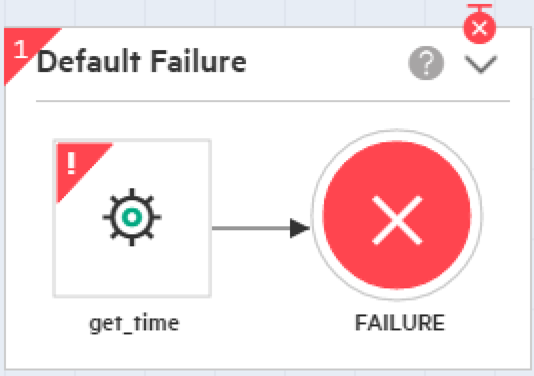
You can set up a default failure step, to run by default for steps that go to the default failure. For example, you might want an email to be sent to the administrator for every failed step.

Every step whose **Failure** response is grounding will run the default failure step before the flows ends in failure.

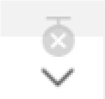
1. Open the flow for which you want to create a default failure step.
2. If the **Default Failure** pane on the canvas is contracted, click the arrow to expand it.



3. Drag an operation or flow from the **Projects** pane or **Dependencies** pane into the **Default Failure** pane.

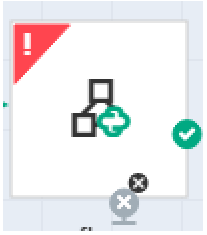


4. Bind the step inputs for this new step (and outputs if needed).
5. Save the flow.

> **Note:** If all the **Failure** response ports are connected to a step, the default failure step is inactive and the icon in the top right corner of the **Default Failure** pane appears in gray.
>
> 

6. To change the default failure step, drag a different operation or flow from the **Projects** pane or **Dependencies** pane into the **Default Failure** pane.
7. To remove the default failure step, select it and click the **Delete**🗑 button from the authoring pane toolbar.
8. To rename the default failure step, select it and click the **Edit** ✎ button in the authoring pane toolbar.

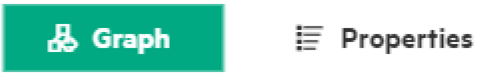
   The FAILURE results in the default error cannot be renamed.

> - If a step is based on a subflow that previously had a **Failure** result step, and this **Failure** result step was deleted, the step based on it will have an upgraded **Failure** result port. If this occurs, click the small x button to remove that result port.
>
>   

> - If a Failure result is added, the step will have a small black grounding icon next to the new failure port, to connect it to **on_failure**.

Define Flow Properties

You can view a flow graph or properties. Use the **Graph** and **Properties** tabs, at the top of the Authoring pane, to switch between the two views.



About the Properties pane

Click **Properties** Properties to display the flow's **Properties** pane. These properties include:

- **Inputs** - flow inputs specify the data required by the flow.

  You define flow inputs when you want to use their value throughout the whole flow and various steps or if you want the inputs to get values from outside (from the user or when other flows use it). Step inputs are the inputs that the contained flow/operation/decision needs.

  If this flow is used as a subflow in another flow, the flow inputs are exposed as step inputs.

  See "Inputs" on page 50.

- **Outputs** - you can set up outputs for a flow.

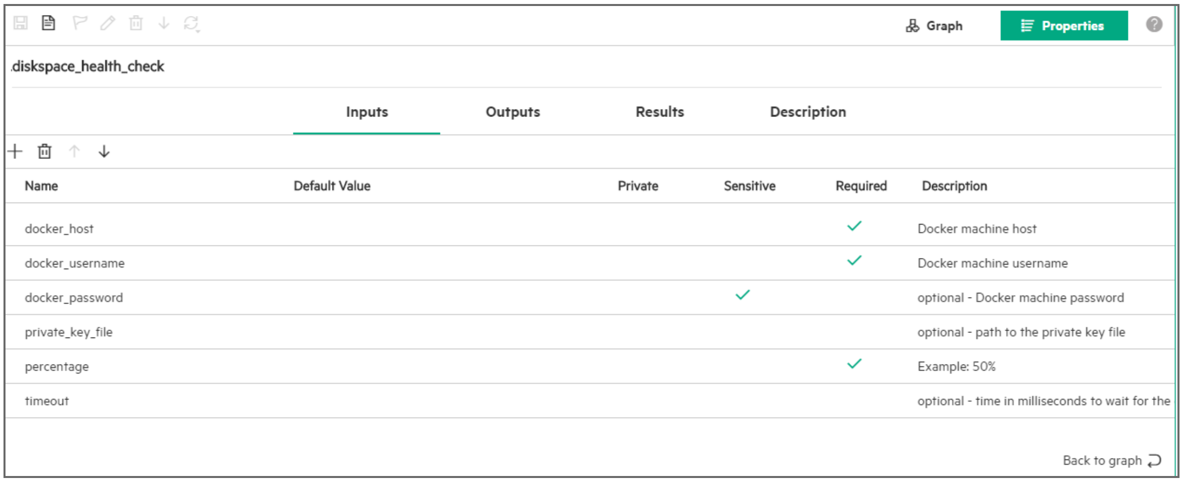
  Once outputs have been set up for the flow, you will be able to choose flow outputs to be published as step outputs when this flow is used as a step.

  See "Outputs" on page 52.

- **Results** - the **Results** tab displays the results that were dragged to the canvas for the flow. You can enter descriptions for these results.

  See "Results" on page 54.

- **Description** - you can enter a summary of what the flow does. It is recommended to include in this summary only the flow level description and not the input/output level description.



Name	Default Value	Private	Sensitive	Required	Description
docker_host				✓	Docker machine host
docker_username				✓	Docker machine username
docker_password			✓		optional - Docker machine password
private_key_file					optional - path to the private key file
percentage				✓	Example: 50%
timeout					optional - time in milliseconds to wait for the

Notes about expressions in OO Designer:

- In the Expression Editor, if there is a multi-line description, each additional line should not start with @. For example, the following is an invalid description:

  ```
  @input input_name: this is the first line
                     @this is the second line
                     this is the third line
  ```

About the Graph pane

The **Graph** pane contains the graphic representation of the flow.

For details, see "Graph Pane" on page 9.

Set up inputs for the flow

Flow inputs are available for the entire flow. For example, you can set the value of a flow variable and use this in various steps in the flow.

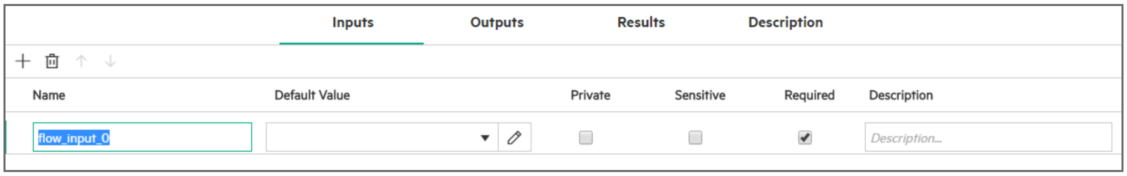
1. Click the **Properties** button ☰ Properties to display the **Flow Properties** pane for the current flow.

   The inputs of the flow are displayed under the **Inputs** tab.

   > **Note:** If you open a flow from the **Dependencies** pane, the **Flow Properties** pane is in read-only mode. If you want to edit the flow, first copy it to the **Projects** pane and edit the copy.

2. To add a new input, click the **New Input** + button.

3. In the new row that appears, enter the details of the input:



Name	Default Value	Private	Sensitive	Required	Description
flow_input_0		☐	☐	☑	Description...

See "Inputs" on page 50.

4. To delete an input, select the input row and click the **Delete** 🗑 button.

5. To move an input up or down in the list, click the **Up** and **Down** ↑ ↓ arrows.

Set up outputs for a flow

Flow outputs are available for the entire flow. A flow output can be used as inputs to other flows.

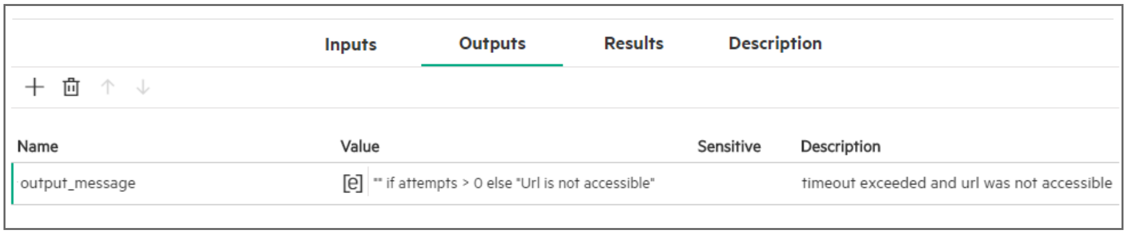
1. Click the **Properties** button ☰ Properties to display the **Flow Properties** pane for the current flow.

   The outputs of the flow are displayed under the **Outputs** tab.

   > **Note:** If you open a flow from the **Dependencies** pane, the **Flow Properties** pane is in read-only mode. If you want to edit the flow, first copy it to the **Projects** pane and edit the copy.

2. To add a new output, click the **New Output** + button.

3. In the new row that appears, enter the details of the output. For example:



Name	Value	Sensitive	Description
output_message	[e] "" if attempts > 0 else "Url is not accessible"		timeout exceeded and url was not accessible

See"Outputs" on page 52.

4. To delete an output, select the output row and click the **Delete** 🗑 button.

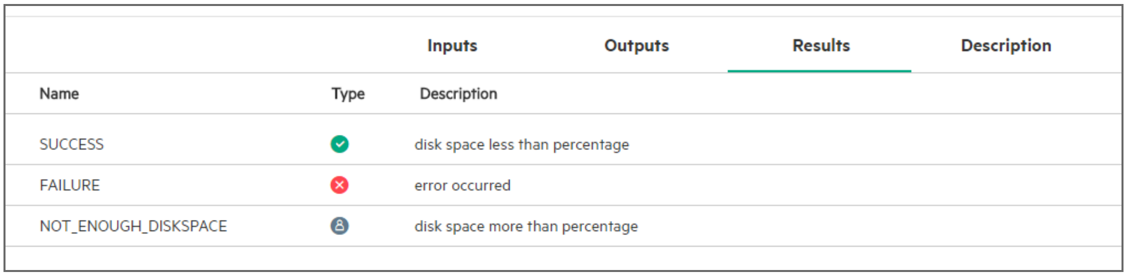
5. To move an output up or down in the list, click the **Up** and **Down** ↑ ↓ arrows.

Adding a description to flow results

On the **Results** tab, you can add description to the results that were added to the flow.

> **Note:** You cannot add new results in this tab. To do that, drag a result step to the canvas from the **Results** toolbar.

1. Click the **Properties** button ☰ Properties to display the **Flow Properties** pane for the current flow.
2. Click the **Results** tab to display the results that have been dragged to the canvas for the flow.



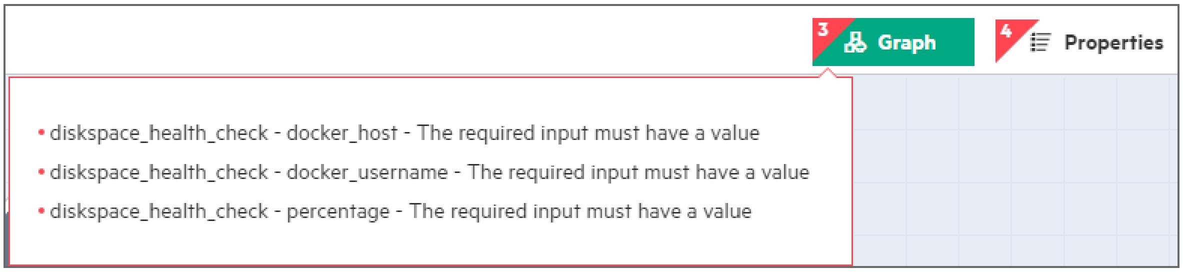
Name	Type	Description
SUCCESS	✓	disk space less than percentage
FAILURE	✕	error occurred
NOT_ENOUGH_DISKSPACE	●	disk space more than percentage

3. (Optional) Enter a description of the results in the **Description** column.

Check for errors in a flow

Flow errors

If a flow or its properties contain errors, an error flag is displayed in the **Graph** and/or **Properties** buttons at the top of the Authoring pane. This flag includes a number, representing the number of errors.



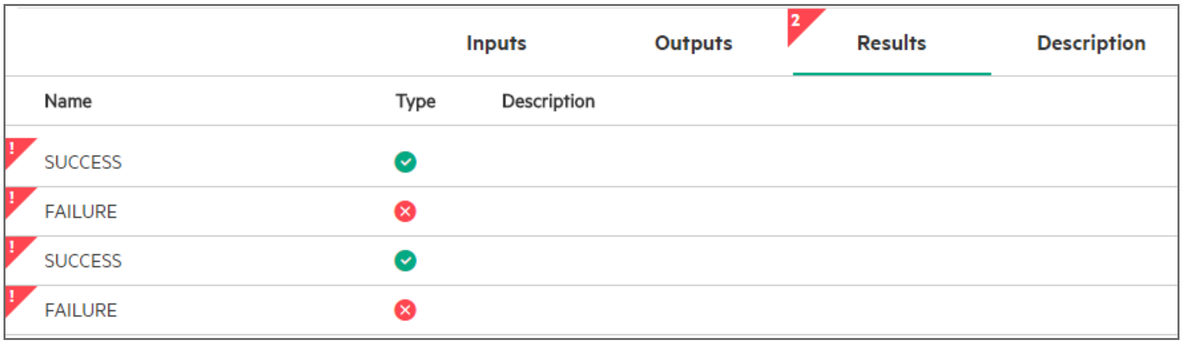
- diskspace_health_check - docker_host - The required input must have a value
- diskspace_health_check - docker_username - The required input must have a value
- diskspace_health_check - percentage - The required input must have a value

Click the error flag to display a tooltip with information about the errors.

After you fix the errors, numbers in the error flag are adjusted, and when all are fixed, the flag disappears.

Error in the flow properties

If errors exist in the flow properties, the error flag appears on the relevant tab and in the rows that are affected. Also at these levels, you can click the error flag to display a tooltip with information about the error.



Name	Type	Description
SUCCESS	✓	
FAILURE	✗	
SUCCESS	✓	
FAILURE	✗	

If the value being entered is not correct from CloudSlang point-of-view, an error will be displayed. If the error causes the corresponding YAML to be corrupted, the value will be reset after removing the focus from the item.

View the CloudSlang code

1. You can see information about the input, output, and result CloudSlang code.

   To view the code defining the flow,operation, decision, or system property, click the **Show as Text** button in the Authoring pane toolbar. A text box appears. This text is the CloudSlang code that will actually run. For details, see "YAML File Textual Representation" on page 90.

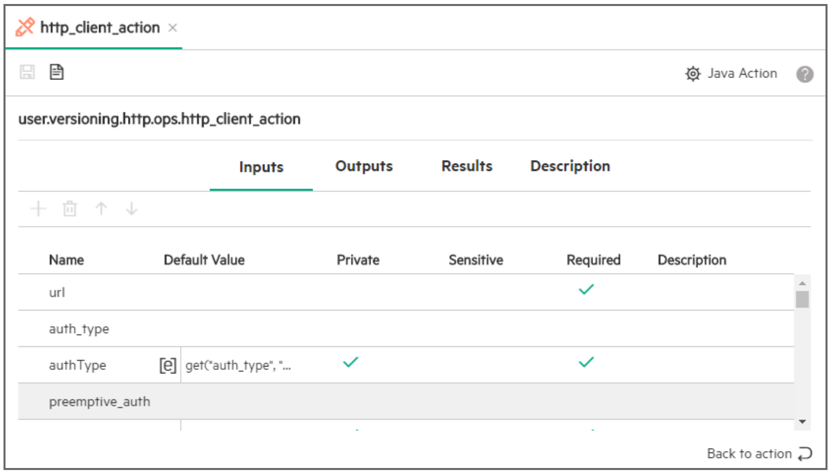
> **Note:** CloudSlang (CloudSlang.io) is an open source environment for process automation, which provides a textual language (YAML over Python), a lightweight engine (which can be embedded in a Java process or run via a command line), and a content library. All the CloudSlang code, including documentation and content, are available in GitHub CloudSlang repositories.
>
> Score is a general-purpose Java-based open-source orchestration engine, which is process-based, embeddable, lightweight, scalable, and multilingual.
>
> The CloudSlang language is a YAML (version 1.2) based language for describing a workflow that can be run by Score engine. The supported file extensions are: .sl, .sl.yaml and .sl.yml.
>
> For more information about CloudSlang, see:
>
> http://www.cloudslang.io
>
> https://github.com/cloudslang/cloud-slang
>
> https://github.com/cloudslang/score

Inputs

Inputs are defined for steps, operations, decisions, or flows.

Under the **Inputs** tab, you can view the inputs that the steps, operations, decisions, or flows receive. The Inputs tab for the steps are on the right side of the canvas. Each input has a value that can be set in a number of ways. It may be a constant value, a python expression, the value of a flow variable or system property, and more.

For example, you can set the value of a flow variable and use this in various steps in the flow.



user.versioning.http.ops.http_client_action

Name	Default Value	Private	Sensitive	Required	Description
url				✓	
auth_type					
authType	[e] get("auth_type", "...	✓		✓	
preemptive_auth					

- **Name.** Displays the name of the input.

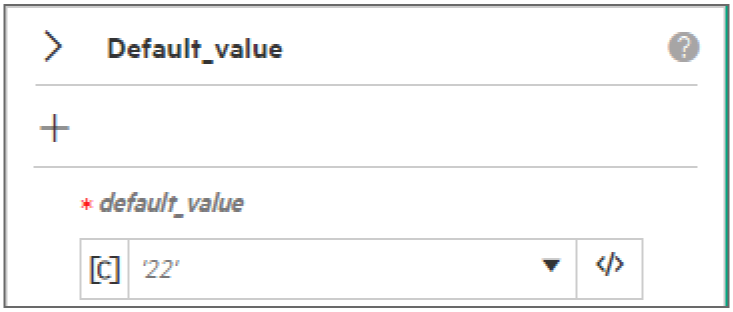
  The combined input name, flow name, and folder path of the input (under the **Library** folder) must add up to no more than 255 characters.

  Input names must start with a letter or an underscore (_) and must contain only the characters (A-Z, a-z, 0-9) or _.
  The Input name cannot contain white spaces.

  > **Note:** For localized characters, this limit is even lower. For example, in Japanese, the 255 characters need to be divided by 4, so the number of characters is only 64.

- **Default value.** A user can add a default value to the input. If no value is passed for this input, the default value is used instead.

  (Optional) You can directly edit the expression of a value. Open the Expression Editor by clicking the **Edit** button next to the value.



For more information, see "Edit the Expression of a Value for an Input or Output" on page 63.

- **Private.** Click the check box if the input needs to be private. This means that it cannot be overridden. This cannot be seen when this flow is triggered or when this flow is referenced as a step

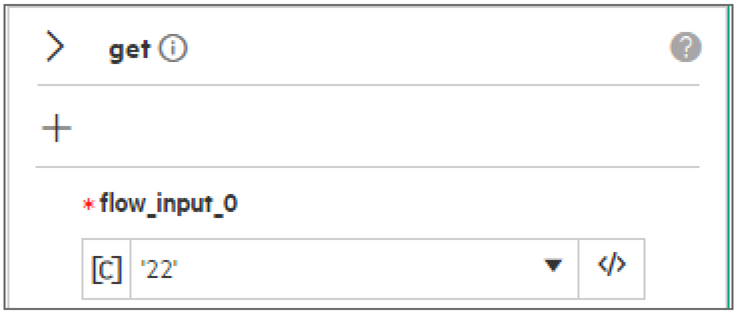
> **Note:** If a flow input is marked as private, it *must* have a default value. If there is no default value, this is flagged as an error.

- **Sensitive.** Click the check box if the input contains sensitive data. During runtime, the value will be encrypted. See "Sensitive Flow Input, Output, and System Properties " on page 53.

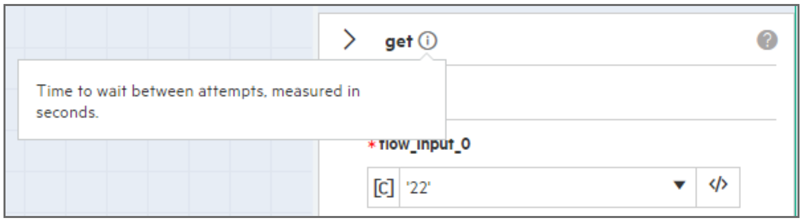
> **Tip:** It is not recommend to set a default constant value when an input is marked as sensitive. The values are displayed as asterisks in the file.

- **Required.** This check box is selected if this input is mandatory for the operation/flow to function.
- **Description.** Displays a description of the input.

When the flow is used as a step (subflow) in another flow, an **Info** ⓘ button will appear when a mouse is rolled over the input (under **Step Properties** > **Inputs**).



When this **Info**ⓘ button is clicked, a smart tooltip will display the description that you entered.



Time to wait between attempts, measured in seconds.

* flow_input_0

[C] '22'

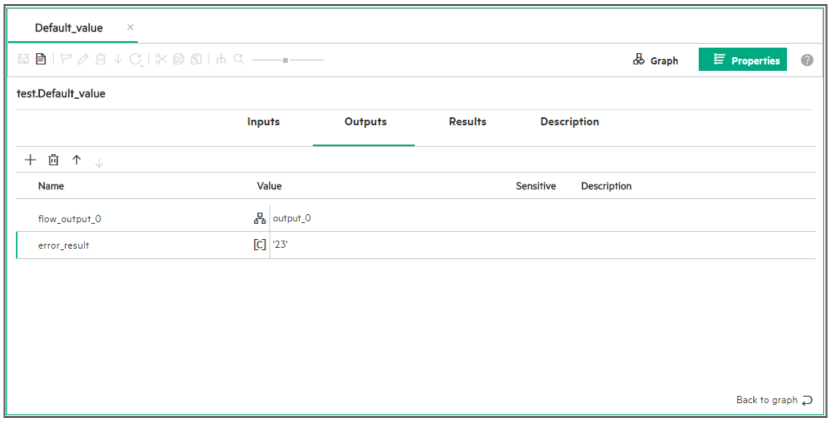
Outputs

Under the **Output** tab, a user can put any information to be send back to the calling flow.

Output values can have complicated expressions, such as trimming, parsing etc. Output expressions must evaluate to strings.

After you set up the outputs for a flow, you can use the flow as a step and add outputs to that step whose value will be taken from the outputs defined in the original flow.

Step Outputs can be used in the flow outputs. For example, if a user adds a step output called 'vacant', the step can be used in a flow output by choosing it from the drop down under the flow variable list

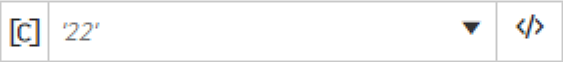


test.Default_value

Name	Value	Sensitive	Description
flow_output_0	output_0		
error_result	[C] '23'		

- **Name.** Displays the name of the output. A default name is provided, but you can change this. Output names must start with a letter or an underscore (_) and must contain only the characters (A-Z, a-z, 0-9) or _.
  The Output name cannot contain white spaces.
- **Value.** Displays how to obtain the flow output value. This may be a constant value, a flow input, published outputs from any of the steps, or any expression.

(Optional) You can directly edit the expression of a value. Click the **Edit** </> button to open the expression editor.



For more information, see "Edit the Expression of a Value for an Input or Output" on page 63.

- **Sensitive.** Select this check box if you want the value to be encrypted at runtime.

  See "Sensitive Flow Input, Output, and System Properties " below.

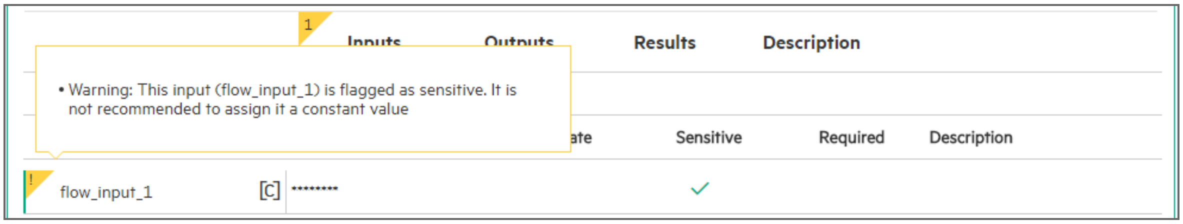
  > **Tip:** It is not recommended to set a constant value when an output is marked as sensitive. The values are displayed as asterisks in the file.

- **Description** Enter a meaningful description of the output.

Sensitive Flow Input, Output, and System Properties

The following can be set as **Sensitive**: an input field, an output field, and a system property.

> **Note:** It is not recommended to set a default constant value when an Input, Output, or System Property is marked as sensitive. The values are displayed as asterisks in the file.

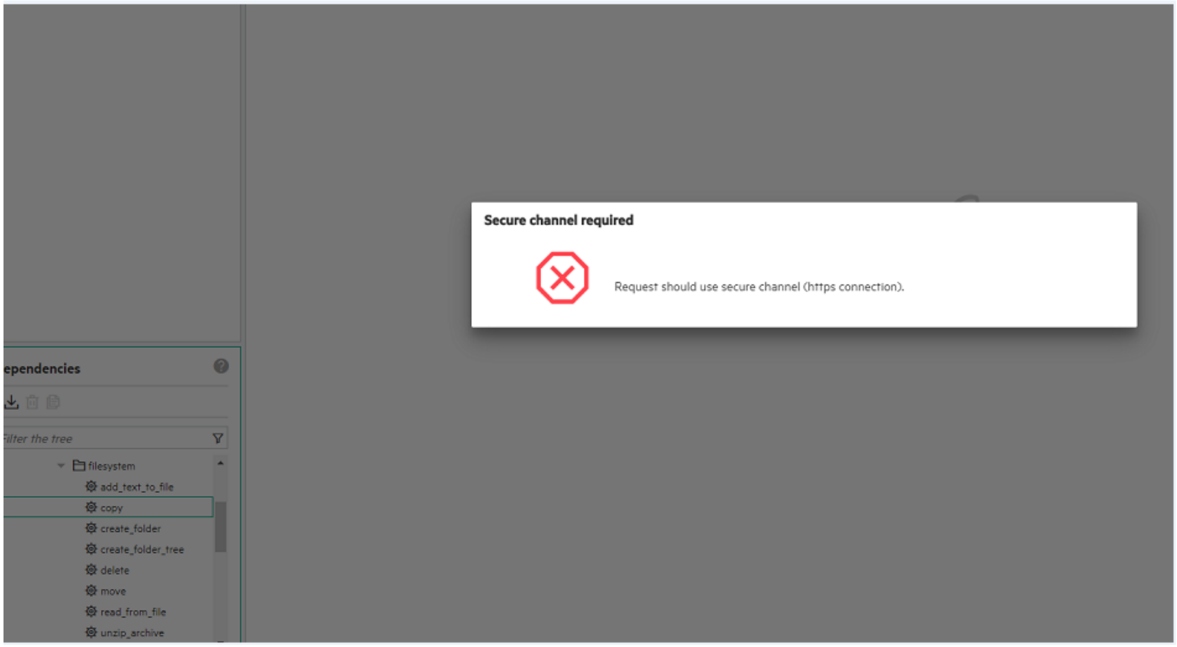


• Warning: This input (flow_input_1) is flagged as sensitive. It is not recommended to assign it a constant value

flow_input_1 [C] ******** ✓

Handle Sensitive Information at Back-end - API

You cannot access the flows and system property files of the content that supports sensitive data through an API using non-secured channel such as HTTP protocol. You are prompted to use secured protocol (HTTPS) as displayed in the below image.



Secure channel required

Request should use secure channel (https connection).

Rules and limitations:

- When an **Input** is defined as **Sensitive**, it flags the system to protect its value when this flow is executed. By extension, any input or expression that refers to it will be treated as **Sensitive**.
- If an **Output** refers to another output that is defined as **Sensitive**, then the resulting **Output** is treated as **Sensitive**.

Results

Under the **Results** tab, view the results that will be returned from the current item.

- **Name.** Displays the name of the result.
- **Type.** Displays the type of the result. The options are **Success** ✓, **Failure** ✗, and **Custom** 🔒.
- **Rule.**Rules can be used to define a result. Rules are relevant only in operations and decisions.
- **Description.** Displays a description of the result.

Define Step Properties

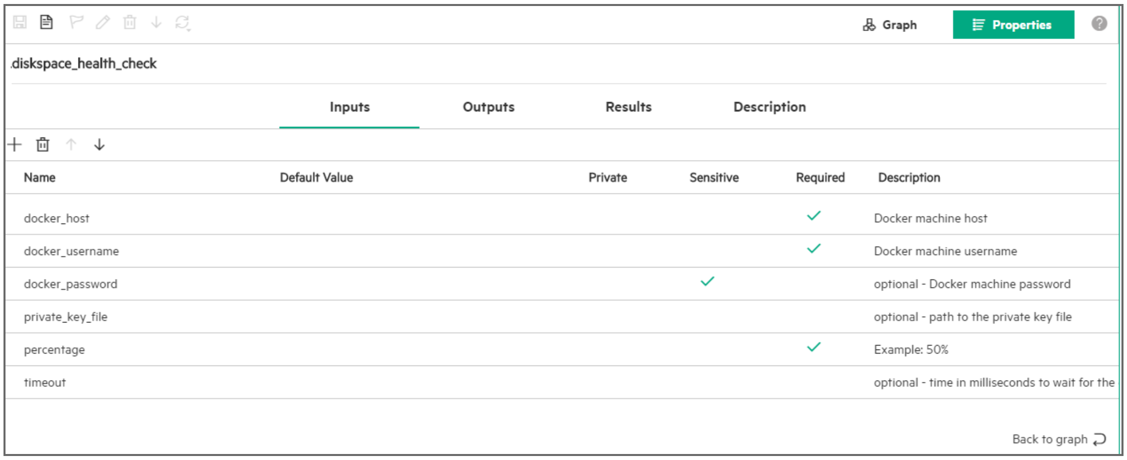
View inputs in a step that was created from another flow

If you include flow inputs in one flow (Flow A) and you use this flow as a subflow step in another flow (Flow B), these inputs are available as step inputs for the step in Flow B.

You can also view the descriptions that were given to the flow inputs in the original flow (Flow A).

1. Create a flow with flow inputs (Flow A), and give these inputs descriptions. For more information about creating flow inputs, see "Define Flow Properties" on page 45.

   For example:



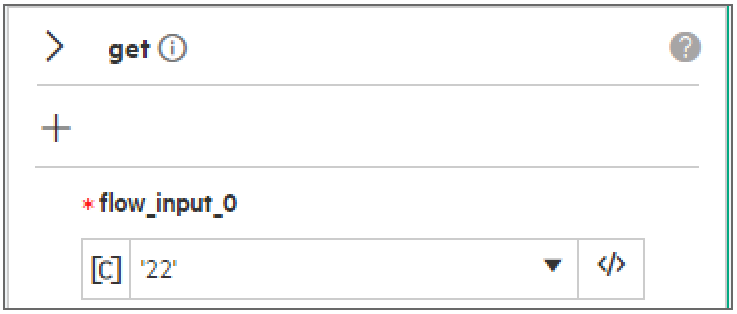
.diskspace_health_check

Name	Default Value	Private	Sensitive	Required	Description
docker_host				✓	Docker machine host
docker_username				✓	Docker machine username
docker_password			✓		optional - Docker machine password
private_key_file					optional - path to the private key file
percentage				✓	Example: 50%
timeout					optional - time in milliseconds to wait for the

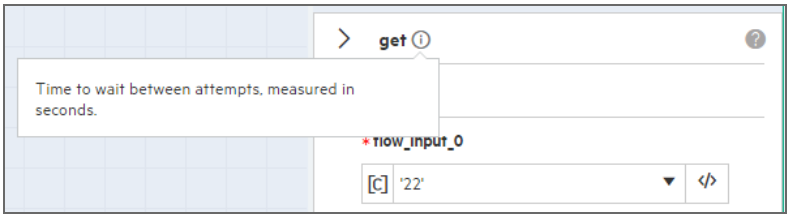
2. Create a second flow (Flow B) and use Flow A as a step.
3. In Flow B, select that step.

   In the **Step Properties** pane, the inputs of the step are displayed under the **Inputs** tab.

4. Roll a mouse over an input, to display the **Info** ⓘ button.



5. Click the **Info** ⓘ button to display a smart tooltip showing the description that was entered for the original flow input.



Time to wait between attempts, measured in seconds.

Add and Edit the Inputs of a step

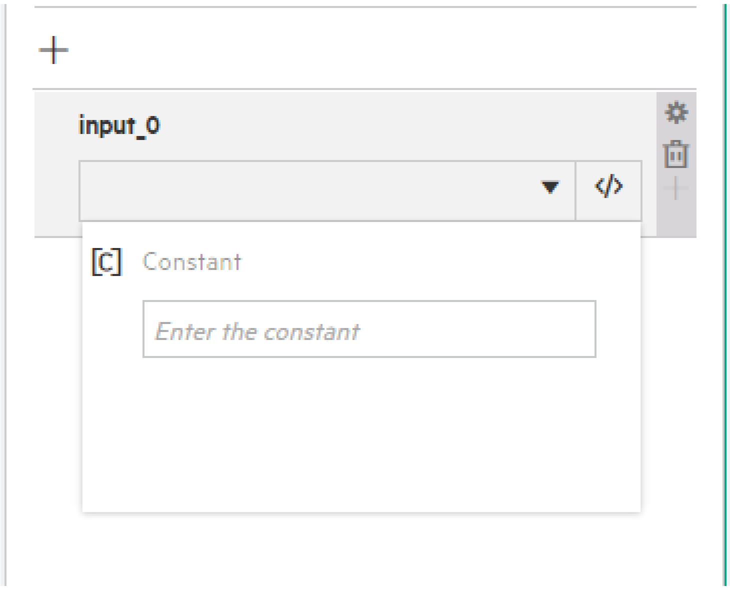
Step inputs are defined for a single step. To add and edit the inputs in a step.

1. Click on the step in the authoring pane. The **Step Properties** pane appears at the right side of the canvas.

   The **Step Properties** pane has two tabs where you can view the inputs and outputs of the selected step.

2. To add a new input for a single step:

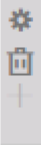
   a. Click the New Step Input + button. The new step input field is displayed.
   For example: `input_0`, `input_1` etc.

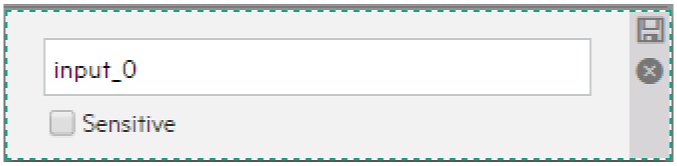
   b. Click the **Value** arrow and enter appropriate values (constant or expression)

   c. Click **Save** 💾 button in the authoring pane to save the step input.

3. To rename a newly added input for a single step:
   a. Hover over the selected step input field to view the step input toolbar as shown below.

   b. Click **Edit** ⚙ icon to enter the editing mode as shown below.

   input_0

   □ Sensitive

   c. Click inside the **Step Input** box and rename as required. For example: From `input_0` to `input_1`

   > **Note:** If you select an existing step input name, a red indicator is displayed warning of a duplicate name.

   d. To set a step input as sensitive, select the **Sensitive** check box.

   > Note Only the custom added step input is editable. The input text box of the step input having a dependency is set to read-only state and is not editable.

   e. Press **ENTER** key or Click **Save** 💾 icon from the step input toolbar to save.

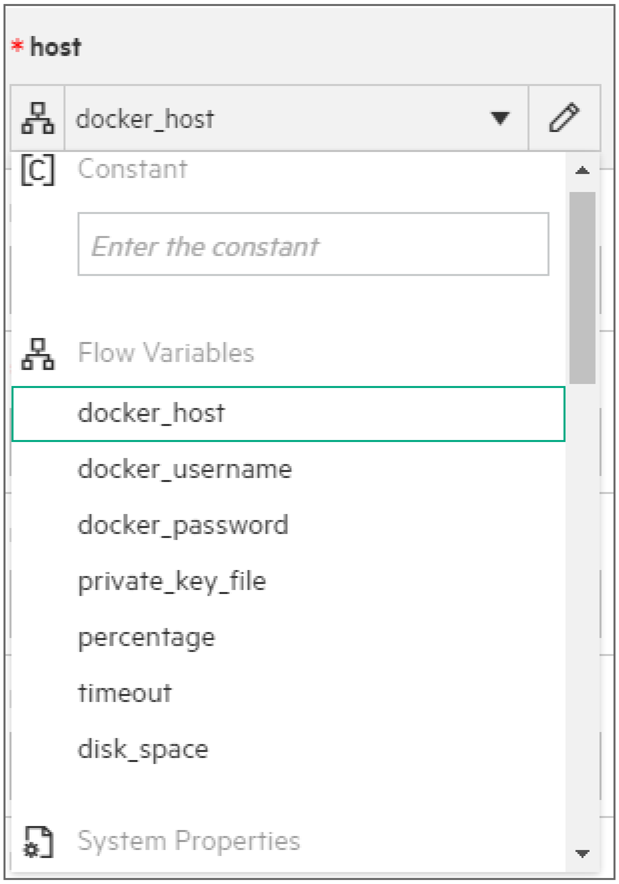
   > **Note** The values of sensitive steps are replaced and displayed with asterisks in both Input field and YAML source. However, the values are cleared when a step input is removed off its sensitive state.

   f. Click **Cancel** ⊗ icon or press **ESC** key to cancel.

4. To edit an input value in the selected step, click the down arrow next to the input value and select a value from the list.

   The default value of the origin flow/operation appears in gray italic text, but you can enter a different value.

   > **Note:** Required inputs appear with an asterisk*.

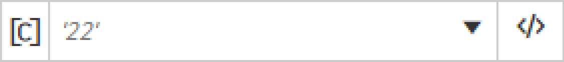


Each input in a step has a value that can be set in the following ways:

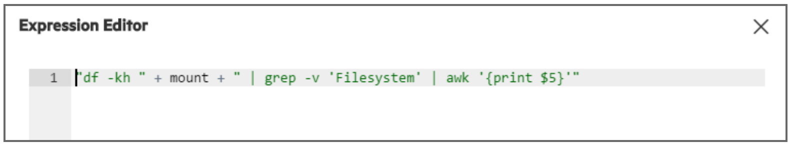
- A **constant value**
- A **default value** as defined in the flow/operation that the step is based on.
- A **system property** - system properties can be defined under **Configuration** > **System Properties**, as described in "Author System Properties" on page 105.
- A **python expression**
- A **flow variable** - flow variables can be defined in the flow inputs, as described in Creating Inputs, Outputs, and Results for a Flow. Flow variables also contain published output data from steps in the workflow.

> **Note:**
>
> - If you choose to define the value from a system property, the content pack or project from which the system property originates is displayed.
> - An empty string value for a required step input is invalid.

5. (Optional) You can directly edit the expression of a value. Open the Expression Editor by clicking the **Edit** ✎ button next to the value.

   In the Expression Editor, enter the value that you want to assign, and click **Save**. The value will appear in the value box without quotation marks.

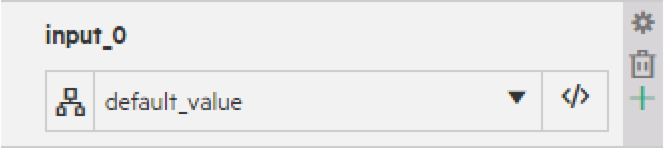
   

   ```
   "df -kh " + mount + " | grep -v 'Filesystem' | awk '{print $5}'"
   ```

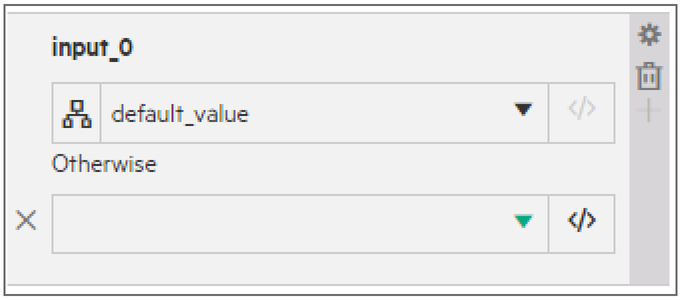
   For more information about the Expression Editor, see "Edit the Expression of a Value for an Input or Output" on page 63.

6. (Optional) If you select a system property or a flow variable, you can add an **Otherwise** option. At runtime, if the flow variable or system property does not exist or is empty, the Expression Editor will use the Otherwise value

   Click the + button to add an Otherwise option.

7. Click the drop-down of the **Otherwise** option and select a value from the list.

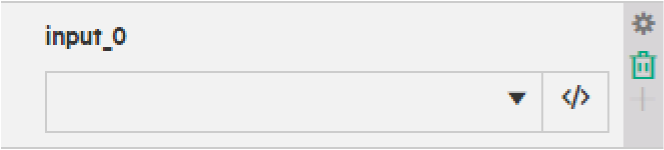
   You can assign the value from a flow variable, a constant, an expression, or a system property.

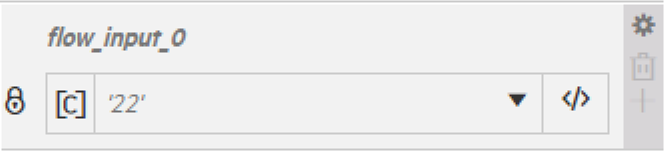
   > **Note:** If you do not click the **New** + button, the **Otherwise** field does not appear and this option remains disabled.

8. To delete an input from a single step:

a. Hover over the selected **Step Input** box to view the step input toolbar.

b. Click the trash 🗑 icon to delete the step input.



> **Note:** Only the custom added step input can be deleted. A step input coming from a dependency is set to read-only state; if it was not edited, and cannot be deleted. Otherwise the input value will be reset to the default value.
>
> 
>

Publish outputs for a step

When setting up step outputs, you choose which information you want to use, from the data that was received in the various flows and operations.

This output will be available to be used as input in other steps and is available to be a flow output:

1. To add outputs to a step, click the step in the Authoring pane. The Step Properties pane appears at the right side of the canvas.

   The outputs of the selected step are displayed under the **Outputs** tab.

2. If the Step Properties pane is hidden, click the **Expand** ‹ button to display it.

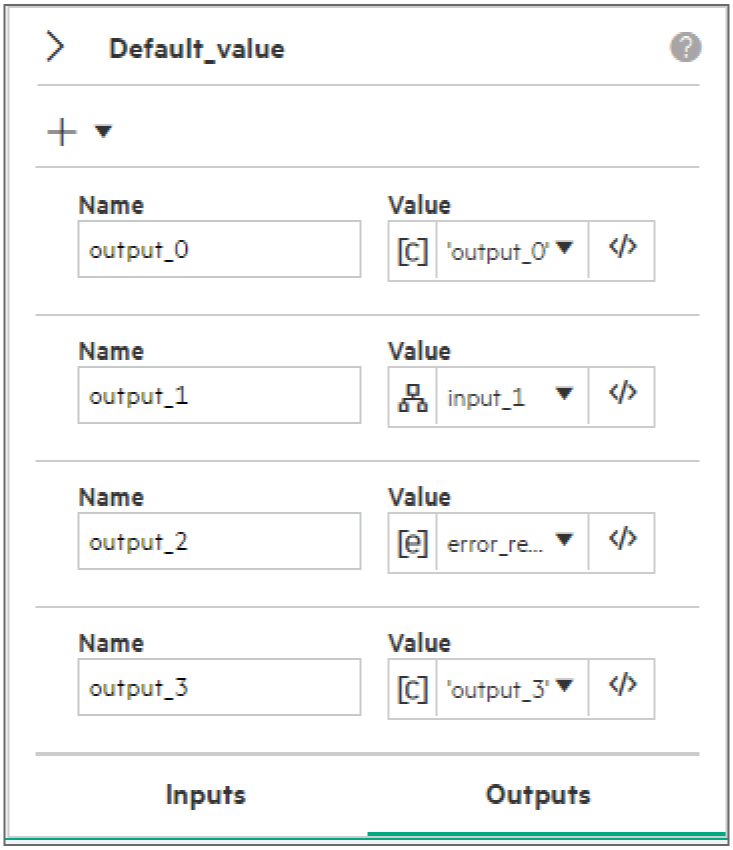
3. To add a new output to the selected step, click the **New** + button or click ⌄ to choose from the list that opens.

4. To edit the name of an output value in the selected step, click in the **Name** field and edit the text.

   > **Note:** While you are entering the output name, if the characters entered cause the underlying YAML file to be corrupted, for example, an empty value, the value is reset to its previous value after focusing out.

5. To edit an output value in the selected step, click the drop-down of the output value and select a

value from the list.



Default_value

Name	Value
output_0	'output_0'
output_1	input_1
output_2	error_re...
output_3	'output_3'

You can assign the value of an output from one of the following:

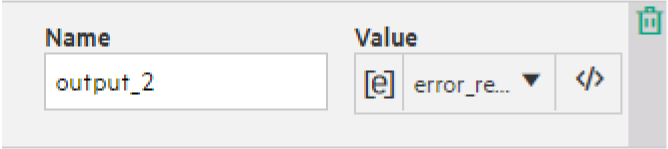
- **Constants.**
- Current step input.
- An **Underline output** - the most common use case is to use outputs received from an operation or flow.
- A **system property** - system properties can be defined under **Configuration**>**System Properties**, as described in "Author System Properties" on page 105.
- Directly edit the expression of a value. Open the Expression Editor by clicking the **Edit** button next to the value. For more information about the Expression Editor, see "Edit the Expression of a Value for an Input or Output" on page 63.

> **Note:** You can extract and modify parts of an output, using Python. See "Extract and Modify Output Values" on page 63.

For example, if you have a step that counts the number of servers that are online, this output

> is published as a flow variable, which can be added to an email that is sent to the system administrator.

6. To delete an output from a step, hover over that output to view the **Delete** 🗑 button, and then click to delete.

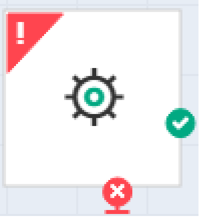


Name: output_2

Value: [e] error_re...

Check for errors in a step

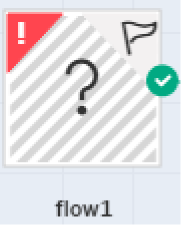
In the Authoring pane, if there is an error in a step operation, for example, missing information, the error appears with a red error flag in the top left corner.



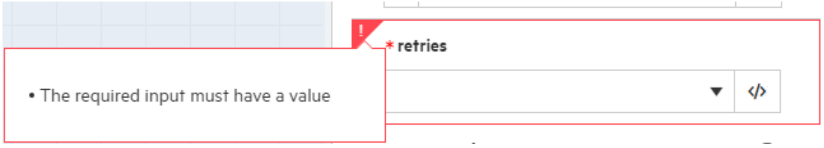
Errors in decisions appear as a red flag covering the top left corner.

> **Tip:** Click the error flag to display a tooltip with information about the error.

If there is a step with a missing dependency (for example, a step that is based on a flow that was deleted), that step appears with a gray, striped background.

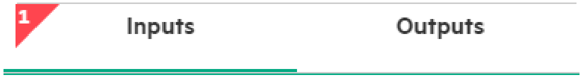


In the **Step Properties** pane, if there are fields with errors in the **Inputs** and **Outputs** tabs, these fields appear with an error flag.



• The required input must have a value

The tabs at the bottom of the **Step Properties** pane also appear with an error flag, which includes a number, representing the number of errors.



When you fix the errors, these error signs disappear.

For more information about errors and invalid content, see "Check for errors in the content" in "Create and Deploy a Content Pack" on page 112.

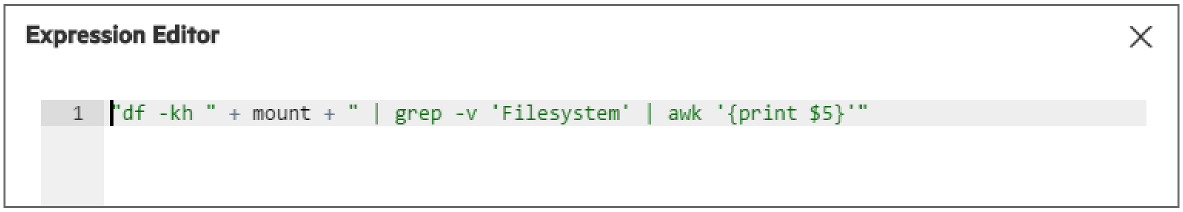
Edit the Expression of a Value for an Input or Output

You access the Expression Editor by clicking the **Edit** ✎ button available from the **Step Properties** pane, on the **Inputs** and **Outputs** tabs, and in the **Flow Properties** pane, on the **Inputs** and **Outputs** tabs.

The Expression Editor displays the python value of the selected input or output value. For example, if you selected a system property named 'sp1' as the input value, this will be displayed as the python expression: `get_sp('sp1')` in the Expression Editor. It can include any valid python expression.

About the Expression Editor

You can directly edit the expression of this value in the Expression Editor.



```
"df -kh " + mount + " | grep -v 'Filesystem' | awk '{print $5}'"
```

Extract and Modify Output Values

You can extract and modify parts of an output, using python.

For example, if you only want the maximum, minimum, and average round-trip times for a ping operation to a server, you can isolate and extract all three pieces of information from the operation's raw output.

Another example is to extract the HTTP code from an HTTP response.

See https://github.com/CloudSlang/cloud-slang-content/tree/master/content/io/cloudslang/base/xml.

Extract and modify the output value

1. In the **Step Properties** pane, under **Outputs**, or in the **Flow Properties** pane, under **Outputs**, select the output and click the Edit button.
2. In the Expression Editor, enter the python expression to manipulate the output value.

   Example: To convert the output to lowercase, you just need to copy the following expression in the OO Designer UI:

   ```
   output1.lower()
   ```

   if the original expression was:

   ```
   pub1: ${output1.lower()}
   ```

3. Click **Save**.

Examples of Using Python to Manipulate Outputs

- **Diff Case**

  Diff Case changes all the characters in the string either to upper case or to lower case.

  Example:

  ```
  # to change to lowercase

  output1.lower()

  # to change to uppercase

  output1.upper()
  ```

- **Extract Number**

  Extract Number extracts the first number found in the result. It treats an unbroken series of integers as a single number.

  Example:

  To extract the number “123” from the strings “123Test” or “Test123”:

  ```
  # extract first number
  # requires another step
      - extract_first_number:
  ```

```
        do:
          strings.match_regex:
            - regex: '\d+'
            - text: ${var_name}
        publish:
          - first_number: ${match_text}
        navigate:
          - MATCH: MATCH
          - NO_MATCH: NO_MATCH
```

- **Format**

  Format attaches text to a result or output or replaces the original content of the result or output with the text that you specify.

  - To attach the text to the front of the existing text, use **prepend**.

    Example:

    ```
    # prepend
    'xyz' + output1
    ```

  - To add the text to the back of the existing text, use **append**.

    Example:

    ```
    # append
    output1 + 'xyz'
    ```

  - To replace the output with the text, use **replace**.

    Example:

    ```
    # replace
    output1.replace('old', 'new')
    ```

- **Line Count**

  Line Count outputs the total number of lines of the result.

  Example:

  ```
  # string line count
  len(output1.split('\n'))
  ```

- **Loop**

  Loop is synchronous. It will act like ordinary loop in other languages (such as java). It runs over a collection of items and executes the step mentioned inside (operation or flow).

- **For**

  The variable name that is assigned the current value from the collection that is defined in the **In** section. This variable is available in the current running of the inner step as flow variable. In case of iterating map there will be two variable names, for the key and value. This variable is required.

  Supported characters: A to Z, a to z, 0 to 9, _ variable name starting with letter or underscore.

  > **Example:**
  >
  > `for: "value in '1','2','3'"`
  >
  > or
  >
  > `for: "value in '1,2,3'"`
  >
  > to actually enter a constant in the expression editor
  >
  > `for: 'k,v in {"key1":"val1","key2":"val2"}'`

- **In**

  The collection to be iterated on. You can use any python expression that results in a list or map. The **In** section contains dropdowns that contain SP and flow variables. Click the Edit button to open the expression dialog where you can insert any string. This variable is required.

  > **Example:**
  >
  > - Array: `['1','2','3','4']` or `['a','b','c','d']`
  > - Python expression: `map(str,range(1,5))` to return a string collection.
  > - Map: `{"key1":"value1","key2":"value2"}`
  > - Comma delimited strings: which would be split automatically into a list: `"1,2,3"`
  > - Flow variable: select it from the drop down (all step published outputs and flow inputs).
  > - System property: select it from the drop down.

- **Break**

  A loop exits when either the collection being looped has been exhausted or the operation or subflow called returns one of the results mentioned in the **break** section. The **break** section is a property of a loop, which is mapped to a list of results on which to break out of the loop. The list of results is a subset of the results of the inner step.

When the operation or subflow of the iterative task returns a result in the break list, the iteration halts and the iterative task navigation logic is run.

If there is no break section in the YAML, by default FAILURE breaks the loop. In Java-based operations and in Python, FAILURE result is added by default to the Break list. You can delete FAILURE.

- **Publish**

  The publish output is set like a regular step (In the step properties). At runtime, it will contain the value from the last iteration of the loop (make sure the 'publish' key is inside the loop).

  You can also aggregate outputs from a loop. To do so, put a python expression that will aggregate the outputs.

  Example:

  The sum is calculated in each iteration and at the end provides the sum of all iterations.

  ```
  - aggregate:
      loop:
        for: "value in '1','2','3'"
        do:
          square:
            - to_square: ${value}
            - sum
        publish:
          - sum: ${sum + squared}
  ```

- **Navigate**

  The navigation logic runs when the last iteration of the step is completed or after exiting the iteration due to a break. The navigation runs according to the last iteration of the loop.

  For keys, the supported characters: A to Z, a to z, 0 to 9.

  For values, restricted to step/result names.

  Example:

  ```
  - custom3:
      loop:
        for: "value in '1','2','3','4','5'"

          do:
            custom3:
              - text: ${value}
          do:
  ```

```
     break:
       - CUSTOM
     publish:
       - CUSTOM: aggregate
       - SUCCESS: skip_this
```

You can also use: `for: "value in '1,2,3,4,5'"`

- **Parallel Loop**

  A Parallel loop runs over a collection in an asynchronous way, meaning that each execution of the step is run simultaneously and in a different context. A new branch is created for each value in a list and the branches run in parallel. The various branches running the flow will run in parallel and the rest of the flow will continue only after all the branches have completed.

  - **For**

    The variable name that is assigned the current value from the collection that is defined in the **In** section. This variable is available in the current running of the inner step as flow variable. In case of iterating map there will be two variable names, for the key and value. This variable is required.

    Supported characters: A to Z, a to z, 0 to 9, _ variable name starting with letter or underscore.

    Example:

    `for: "value in '1','2','3'"`

    or

    `for: "value in '1,2,3'"`

    to actually entering a constant in the expression editor.

    `for: 'k,v in {"key1":"val1","key2":"val2"}'`

  - **In**

    The collection to be iterated on. You can use any Python expression that results in a list or map. The **In** section contains drop downs that contain SP and flow variables. Click the Edit button to open the expression dialog where you can insert any string. This variable is required.

    Example:

    - Array: `['1','2','3','4']` or `['a','b','c','d']`
    - Python expression: `map(str,range(1,5))`
    - Map: `{"key1":"value1","key2":"value2"}`

- Comma delimited strings: which would be split automatically into a list: `"1,2,3"`
- Flow variable: select it from the drop down (all step published outputs and flow inputs).
- System property: select it from the drop down.

- **Publish**

  The publish mechanism defines the step aggregation logic, generally making use of the **branches_context** construct.

  The branches context is a list that includes all the contexts of all the iterations. It is set according to the order of execution, meaning that the execution that finishes first gets the first place in the list and so on.

  After all branches of a Parallel Loop have completed, the execution of the flow continues with the publish section. The expression of each output pair is evaluated and published to the flow scope.

  In Parallel Loops, the publish keyword is at the loop level in the YAML and not as in a Loop where it is inside the loop, in line with the **do** keyword.

  In a Parallel Loop, there is no publish section inside the inner step. Instead, it publishes all the outputs of the inner step by default to the **branches_context**, allowing for aggregation.

  The aggregation syntax should be a python expression.

  Example:

  - Retrieves the name variable from the first branch to finish:

  ```
  - publish:
    - first_name: '${str(branches_context[0]["name"])}'
  ```

  - Publishes an aggregate output (it iterates over the :', which contains a list of all the iterations of the Parallel Loop, and takes the value of 'name' from each branch):

  ```
  - publish:
    - name_list: '${str([str(x["name"])
          for x in branches_context])}'
  ```

  - Creates a list of branch results (as branches of a Parallel Loop complete, branch results get placed into the branches_context list under the branch_result key):

  ```
  - publish:
    - result_status_list: '${str([str(x["branch_result"])
          for x in branches_context])}'
  ```

- **Navigate**

The navigation logic runs after the last branch has completed. If any of the branches return a result of FAILURE, the flow will follow the navigation path of FAILURE. Otherwise, the flow will follow the SUCCESS navigation path. Any other result beside SUCCESS and FAILURE will be evaluated as SUCCESS.

For keys, the supported characters: A to Z, a to z, 0 to 9.

For values, restricted to step/result names.

> - If the inner step does not contain a FAILURE result, the navigation (ports) do not contain a FAILURE port, but only a SUCCESS port.
> - If a step has a custom result, after setting it in a parallel loop, the ports change to be SUCCESS and FAILURE. After removing the step from the parallel loop, the original ports of the step is returned.

Example:

```
- print_values:
    parallel_loop:
       for: value in values
       do:
          print_branch:
             - ID: ${value}                                                     do:
    publish:
       - name_list: '${str([str(x["name"])
            for x in branches_context])}'
    navigate:
       - SUCCESS: print_list
       - FAILURE: FAILURE
```

- **Regular Expression**

  Regular Expression filters the raw results using a regular expression (regex).

Example:

```
# regular expression
# as above, requires another step
base.string.match_regex:
  - regex: '<expression>'
  - text: ${output1}
```

- **Replace**

  Replace replaces the first or last instance or all instances of one string with another string.

  - Use **First**, **All**, or **Last,** to define which instances of the target string to replace.

    Example:

    ```
    # replace first
    output1.replace('old', 'new', 1)
    ```

  - To make the search not case-sensitive, use **Ignore case**.

  Example:

  ```
  'abcabdAb'.lower().replace('ab'.lower(), 'XX')
  ```

- **Round Number**

  Round Number rounds numbers up or down.

  - Use **Number of Decimal Places** to define the number of decimal places the number should be rounded to.

    Example:

    ```
    # round
    # note: float function call is not always needed
    # note: python rounding of floats sometime returns
    # unexpected results
    # see https://docs.python.org/2/library/functions.html#round
    round(float(output1), <number_of_decimal_places>)
    ```

  - **Round** rounds the number up if the last significant digit is 5 or more, and down otherwise

  Example:

  ```
  round(2.734) -> 2.73

  round(2.734, 2) -> 2.73
  ```

- **Select Range**

  Select Range defines a string that you want to extract from the input data. The two criteria for defining the string are its length in characters and the position of its first character from the start of the input data.

  - Use **Start** to specify the zero-based start position of the string.
  - Keep in mind that a new line may count as one or two characters, depending on the operating system from which you obtain the data you're filtering.

Example:

```
# select range

# note: <end> is not included in the selection

output1[<start>:<end>]
```

- **Sorted**

  Sorted orders the lines of input data by the first character in each line.

  - To specify the direction of the sort, use **Ascending**.
  - To order the data by ASCII order, use **Treat as Numbers**.

    Note that ascending ASCII order is roughly as follows, for English characters:

    - White space
    - Symbols
    - Numbers
    - Alphabetic characters

Example:

```
# sort lines
# note: returns a list, can be joined to a single string using join function
sorted(output1.split('\n'))
```

- **Strip**

  Strip strips characters from the beginning or end of the raw results.

  - Use **Strip Method** to specify how you want the filter to strip the raw results:
    - **All Characters Up To** the string.
    - **All Characters Up To And Including** the string.
    - **All Characters After** the string.
    - **All Characters After And Including** the string .
  - Use **Characters to Strip** to specify the string to find.

Example:

```
# strip other characters
# note: can also be used with lstrip and rstrip
output1.strip(<characters_to_strp>)
```

- **Strip Whitespace**

  Strip Whitespace removes all the whitespace characters from the front and the end of the raw results.

  > **Example:**
  >
  > ```
  > # strip whitespace from both sides
  > output1.strip()
  > # strip whitespace from left side
  > output1.lstrip()
  > # strip whitespace from right side
  > output1.rstrip()
  > ```

- **Table**

  Table does not convert the raw results into a table, but enables you to manipulate the raw results as if they were a table, including sorting columns and selecting columns, rows, and blocks.

  - Use **Column Delimiter** to specify the character that will serve to divide the data into columns in a meaningful way.
  - Use **Row Delimiter** to specify the character that will serve to divide the data into rows in a meaningful way.

    > **Note:** Two or more consecutive whitespaces count as a single whitespace, so a column may be occupied by data that you expected to find in a column to the right. For example, this behavior will appear if you apply this filter to the output of a “dir” command-line command with whitespace specified as the column delimiter.

  - Use **First Row is Header** to treat the members of the first row as column headers, .
  - Use **Strip First Row of Result** to remove the first row.
  - Use **Sort On Column** to sort on a column, by specifying the (1-based) column number.

    > **Tip:** The value **-1** means do not sort on any column.

  - Use **Ascending** to specify the ascending order. By default, the sort order is descending.
  - To select the row that you want the filter to extract, specify the row number with **Select Row** (**-1** selects all the rows) and specify the number of columns in that row that you want extracted using **Select Width** (**-1** selects all the remaining columns to the right of the column specified by **Select Col**).
  - To select the column that you want the filter to extract, specify the column number with **Select Col** (**-1** selects all the columns) and specify the number of rows in that column that you want

extracted using **Select Height** (**-1** selects all the remaining columns to the right of the column specified by **Select Row**)

Example:

To extract the first 5 rows of the 2nd through 4th columns, you would specify the following:

```
Select Row= 0
Select Height= 5
Select Col= 2
Select Width= 3
```

In these settings, the first two settings define the selected rows, and the last two settings define the selected rows.

- **XML**

  XML enables you to parse XML within a step, the XML being obtained from the step's input or output. The **XML Get Attribute** filter extracts the value for each of one or more instances of the attribute that you specify. You can control which instance of the attribute the filter is applied to by specifying an element path to the attribute.

  You can obtain the value for a single instance of the attribute or for multiple instances, returned in a table. In such a table, the columns are comma-delimited and the rows are new line-delimited.

  - Using **Element Path**, specify the path of the element that contains the attribute whose value you want to extract. Use forward slashes to separate the parts of the path to the element.

    To control which instance of the element the filter gets the attribute's value from, add a specification such as [2] or [3]. The numbering of elements is 1-based (starts with [1]). Thus to specify the second instance of an element, you would use [2].

  - To search child elements of the element you've specified, use **Include sub-elements**.
  - Using **Attribute Name**, specify the name of the attribute whose value you want.
  - For **Result**, use one of the following:
    - To restrict the result extracted to the value of a single instance of an attribute, use **Single Match**.
    - To extract the value of all the instances of the attribute you have named, use **As Table**.

Example:

```
# xml
# requires another step
base.xml.xpath_query:
  - xml_document: ${output1}
  - xpath_query: <query>
  - query_type: <node|value|nodelist>
```

- **XML Get Element**

  XML Get Element enables you to extract an element in its entirety (including child elements, values, and attributes) by describing it in any of the following ways:

  - Use **Element Path** to specify an absolute path to the element.

    - **../** specifies the parent of the last-named element.
    - **./** specifies the last-named element.

    You can specify a particular instance of any element in the path with an integer inside square brackets.

    Example:

    `/tickets/ticket/details/comment` specifies all the comments in the details for all the tickets.

    `/tickets/ticket/details/comment[2]` specifies the second comment for each ticket.

    `/tickets/ticket[2]/details/comment` specifies all the comments for the second ticket.

  - Use **Child Named** to specify the name of an element that is a child of the element (or elements) that you want to extract. If the child element has a value, you can narrow the results by specifying that value using **Value**.

    - **Child Named** works for only one level of child elements. The filter only returns the direct parent of the child element that you specify.
    - **Value** is intended for brief values. The value that you type there must be an exact match of the value of the child element of the element that you want to extract.

  - Using **Attribute Named**, specify the name of an attribute that is unique to the element you want to extract. To further narrow the results, you can give a value of the attribute for **Value**.

  Example:

  ```
  # xml
  # requires another step
  base.xml.xpath_query:
    - xml_document: ${output1}
    - xpath_query: <query>
    - query_type: <node|nodelist>
  ```

- **XML Get Element Value**

  XML Get Element Value enables you to get the value of a specific element.

  Use **Element Path** to specify the path to the element whose value you're interested in.

  As with the other filters, if there are multiple instances of an element, the filter returns the first one, unless you specify a different instance.

  Example:

  ```
  # xml
  # requires another step
  base.xml.xpath_query:
    - xml_document: ${output1}
    - xpath_query: <query>
    - query_type: <node|value|nodelist>
  ```

- **XPath Query**

  XPath Query enables you to extract data from the result with queries that use the standard XPath syntax, which you define for **XPath Query** .

  - The path that precedes the square brackets identifies the scope of the query with which you are narrowing the results.
  - Square brackets contain the filtering portion of the query. There can be more than one set of filters in a query.

  Example:

  ```
  # xml
  # requires another step
  base.xml.xpath_query:
    - xml_document: ${output1}
    - xpath_query: <query>
    - query_type: <node|value|nodelist>
  ```

Add Decisions to a Flow

A Decision is a step that serves as a gateway for a decision in a flow. A Decision performs a calculation and then controls the direction of the flow, based on rules that are used to calculate the result. Decisions do not perform actual operations but only calculate where to navigate next.

More about decisions

For example, a Decision might compare two values and return the difference between them. If the result is higher than a specified value, the flow will branch to one direction and if the result is lower, it will branch to another direction.

It is optional to add Decisions to a flow.

Decision names can contain alphanumeric characters, and underscores (_).

Decision names cannot contain parentheses (()), square brackets ([]), curly braces ({}), hypens (-), spaces ( ), and dots (.).

Decisions are read-only and cannot be created and edited in OO Designer, or copied to the **Projects** pane.

You can use them as steps in flows by dragging them from the **Dependencies** pane.

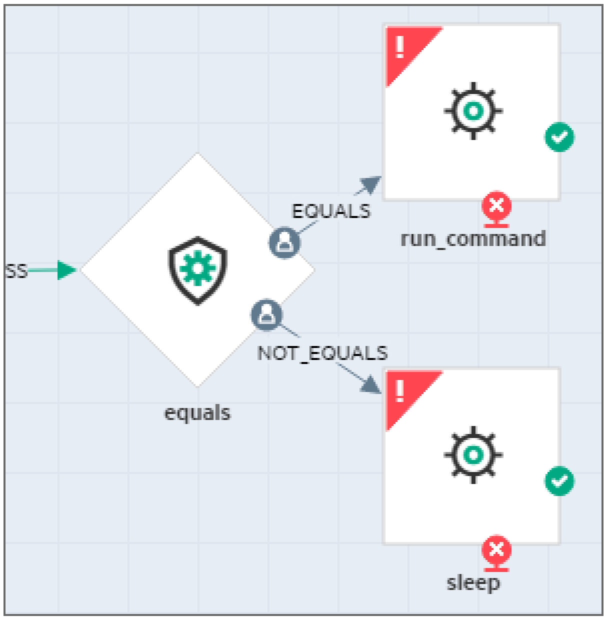
Add a decision step to a flow

1. Drag a Decision from the **Dependencies** pane to a flow in the **Authoring** pane.

   Decisions appear in the **Dependencies** pane with an icon in the shape of a badge .

   In the **Authoring** pane, they appear in the shape of a diamond.

2. Drag navigation lines from the decision step's response ports to other steps.



3. Click the decision step to open the **Step Properties** pane at the right side of the canvas.

4. Under **Inputs**, enter any relevant inputs. For example, if the action is to compare two values, specify how these values will be assigned to the inputs.

   For more information about assigning values to inputs, see "Define Step Properties" on page 55.

5. Under **Outputs**, enter any relevant outputs.

View the details of a Decision

You can view their details in the Editor.

1. Double-click a **Decision** in the **Dependencies** pane, in order to display its details in the Editor.

   Decisions appear in the **Dependencies** pane with an icon in the shape of a badge .

2. Under **Inputs**, view the inputs that were defined for this decision. For example, if the action is to compare two values, these are included as inputs.

   See "Inputs" on page 50.

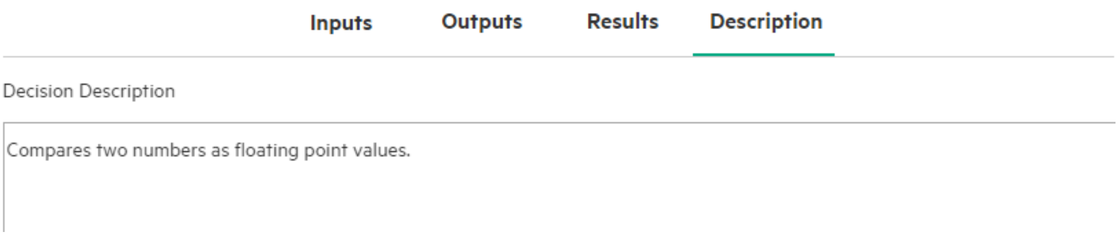
3. Under **Outputs**, view any outputs that were defined for this decision.

   See "Outputs" on page 52.

4. Under **Results**, view the results that were defined for this decision.

   See "Results" on page 54.

5. Under the **Description** tab, view a description of what the Decision does, if one was entered.

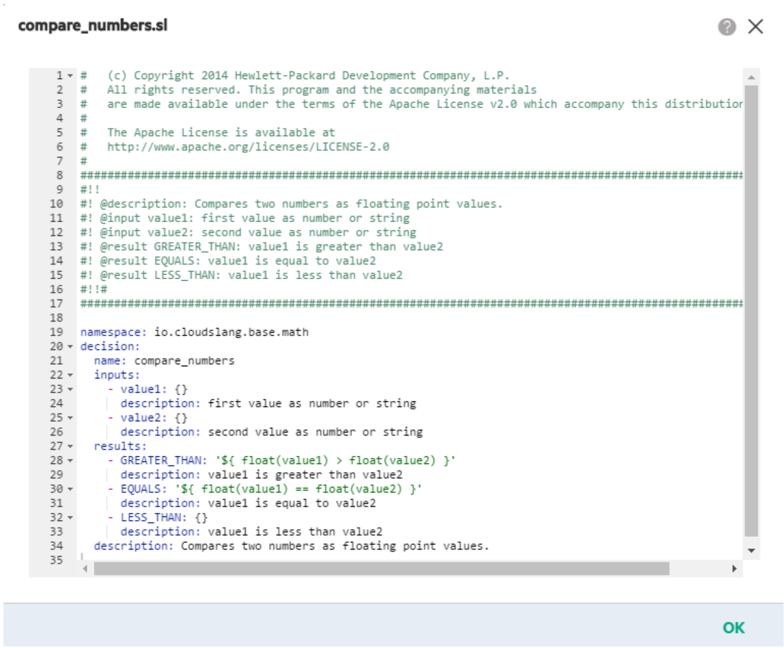


Decision Description

Compares two numbers as floating point values.

6. To view the code behind the Decision, click the **Show as Text** button. A text box appears,

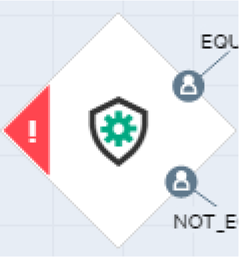
displaying the code.



compare_numbers.sl

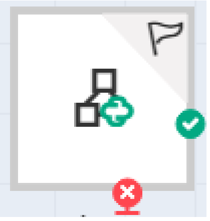
```
#   (c) Copyright 2014 Hewlett-Packard Development Company, L.P.
#   All rights reserved. This program and the accompanying materials
#   are made available under the terms of the Apache License v2.0 which accompany this distributio
#
#   The Apache License is available at
#   http://www.apache.org/licenses/LICENSE-2.0
#
##################################################################################################
#!!
#! @description: Compares two numbers as floating point values.
#! @input value1: first value as number or string
#! @input value2: second value as number or string
#! @result GREATER_THAN: value1 is greater than value2
#! @result EQUALS: value1 is equal to value2
#! @result LESS_THAN: value1 is less than value2
#!!#
##################################################################################################

namespace: io.cloudslang.base.math
decision:
  name: compare_numbers
  inputs:
    - value1: {}
      description: first value as number or string
    - value2: {}
      description: second value as number or string
  results:
    - GREATER_THAN: '${ float(value1) > float(value2) }'
      description: value1 is greater than value2
    - EQUALS: '${ float(value1) == float(value2) }'
      description: value1 is equal to value2
    - LESS_THAN: {}
      description: value1 is less than value2
  description: Compares two numbers as floating point values.

```

If there is an error in the decision step, a red error flag is displayed. Click the flag to display a tooltip with information about the error.



Change the Start Step

By default, the first step that is dragged onto the canvas for a flow is flagged as the start step for that flow. The start icon appears in the corner of the step.

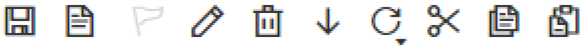


However, you can assign another step to be the one that starts the flow.

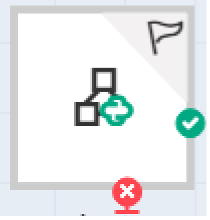
Change the start step in a flow

To assign a step as the starting point of a flow:

- Right-click the step and choose **Set as start Step** option from the list or
- Select the step, and then click the **Set as Start Step** icon in the Authoring pane



The start icon appears in the corner of the step icon, and is removed from the default start step.



Remove the start step in a flow

It is not possible to have a flow with no start step.

The only way to remove the start icon from a step is to assign a different step as the start step.

Create Flows with Loops

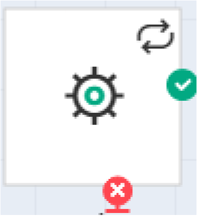
You can set a step as a loop, in order to create an iterative task. A loop step can be based on an operation or flow.

OO Designer supports two kinds of loops:

- **Loop**

  A loop step runs over a collection of items in a **synchronous** way.

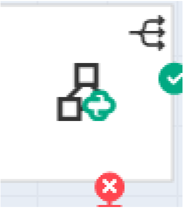
  Steps that have been set as a loop appear with a circular arrows icon.



- **Parallel loop**

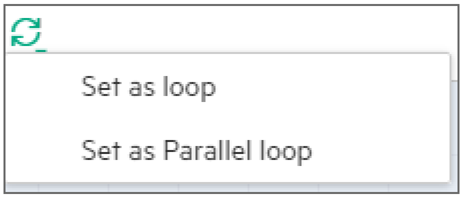
  A parallel loop step runs over a collection of items in an **asynchronous** way. This means that the step can run simultaneously and in different contexts.

  Steps that have been set as a parallel loop appear with a branch icon.

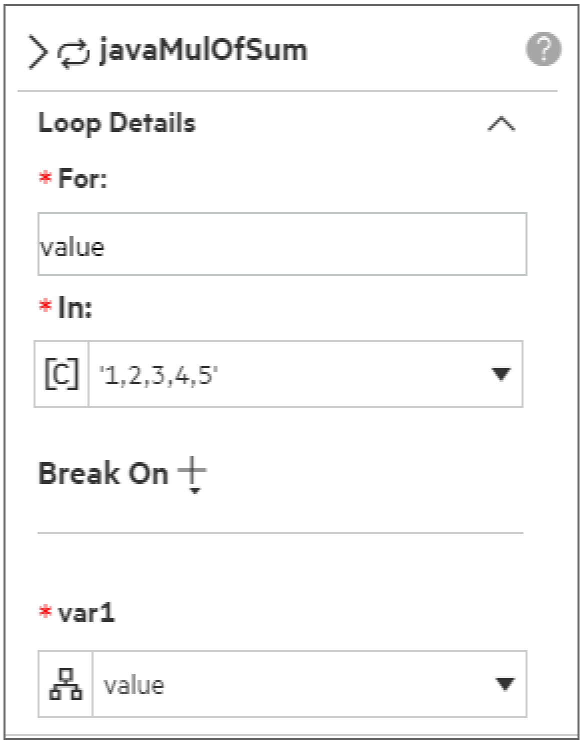


Create a flow with a loop

1. Create a flow and select the step that you want to use as a loop step.
2. Click the **Create Loop** 🔄 button in the authoring pane toolbar.
3. From the menu that appears, select **Set as loop**.



4. For a step that has been set as a loop, the **Step Properties** > **Inputs** tab has an extra **Loop Details** section, where you can define the collection that the input will be iterated on and the break points of the loop.



javaMulOfSum

Loop Details

* For:

value

* In:

'1,2,3,4,5'

* var1

value

- **For**: Enter the name of the variable that will be assigned the current value from the collection that is defined under **In**. This step is compulsory.

  This variable will be available while the loop step is running.

- **In**: Define the collection to be iterated on. This step is compulsory.

  You can define the collection using a flow variable or system property. For example, you can create a system property with a comma delimited string ('1,2,3,4,5') as the value and use this to define the collection.

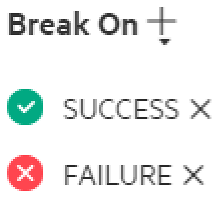
  > **Note:** The evaluation of the flow variable or system properties must be a string as CloudSlang supports only strings. For example. 'Range(1,6)' will work inline only if the **for** variable (which will be a digit) isn't assigned to an inner step input (which requires string). It will not work if passed as a value of system property or flow variable.

  You can also open the Expression Editor and write a Python expression that results in a list or map. For example:

  - An array: ['1','2','3','4'] or ['a','b','c','d']
  - A Python expression: map(str,range(1,5))

    - A map: {"key1":"value1","key2":"value2"}
    - A comma delimited string, which will be split automatically into a list: '1,2,3'.
  - **Break On**: A loop ends either when the list that it is looping on is finished or when the looped step returns one of the results defined as a break point.

    Click the **Select break result to add** + button to choose one of the step results as a break point. There can be multiple break points.

    > **Note:** To remove a break point, roll over it so that the **Remove** × button is displayed, and click the **Remove** × button.
    >
    > 
    >

5. In the **Step Properties** > **Outputs** tab, define how the output will be published. At run time, the output will contain the value from the last iteration of the loop.

   a. Click the **Add a Step Output** + button to add a new output to the step.

   b. Choose one of the underline outputs to define the output.

   c. Under **Value**, open the Expression Editor and enter a Python expression to aggregate all the iterations.

   d. Set up the navigation logic for the loop step.

   The navigation to the next step in the flow occurs when the last iteration of the step is completed or after the loop exits the iteration due to a break. The navigation will run according to the last iteration of the loop.

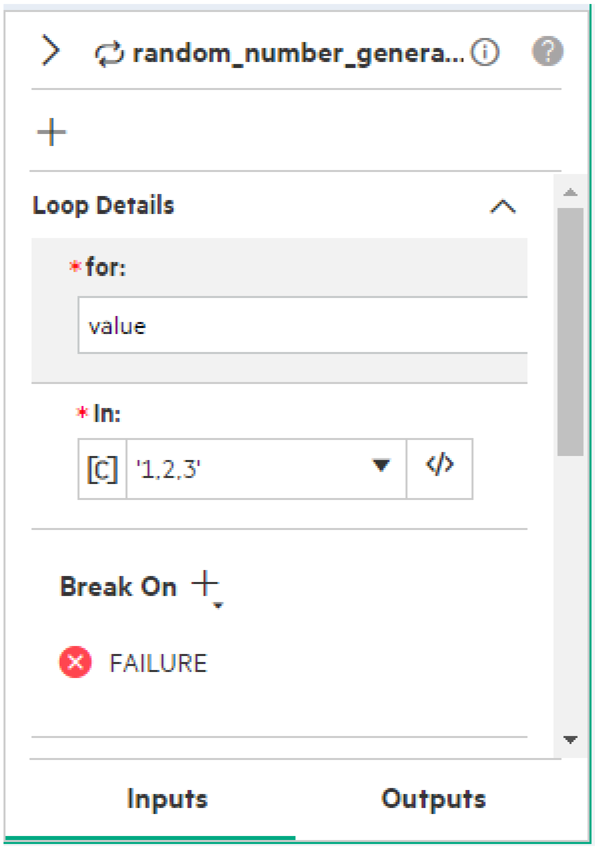
   > **Note:** If you want to use the **for** value inside the step, you must add it as the value to one of the step inputs.

Aggregation of Loop Outputs

To aggregate the results of a specific loop step, proceed as in the following example, where we add the numbers generated in the loop's iterations:

1. Import a base content pack onto dependencies pane and choose the **random_number_generator** operation.

2. Drag the **random_number_generator** operation to the flow and set the resulting step as a loop.
3. Define the collection that the loop will run on as shown in the following example image.



random_number_genera...

Loop Details

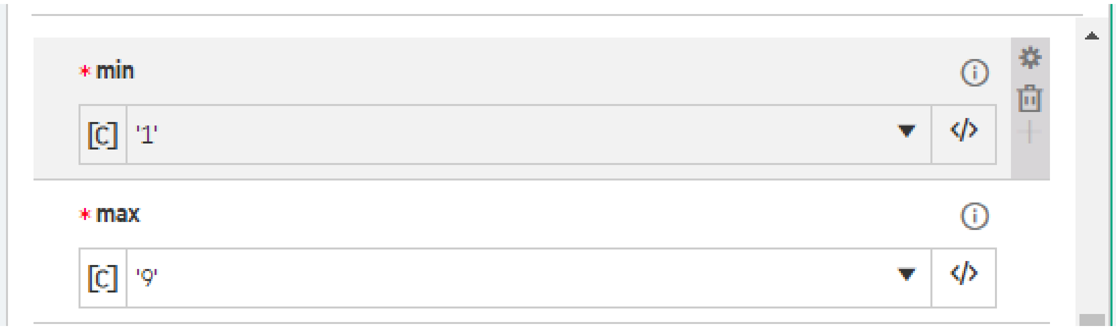
*for: value

*In: [C] '1,2,3'

Break On

FAILURE

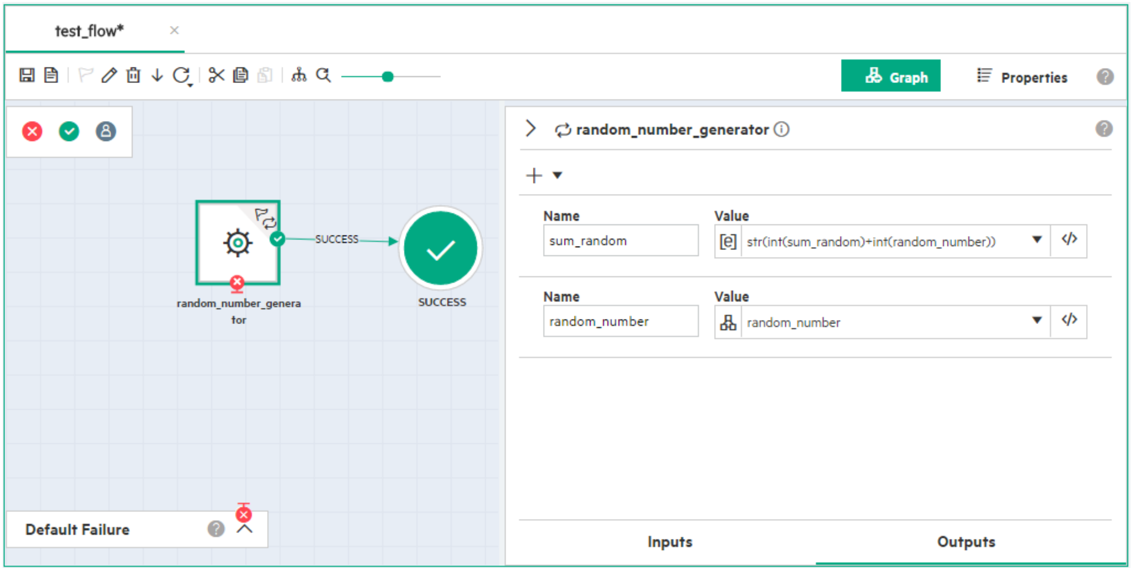
4. Enter the values for **min** and **max** Step Inputs as shown in the following example image.



*min [C] '1'

*max [C] '9'

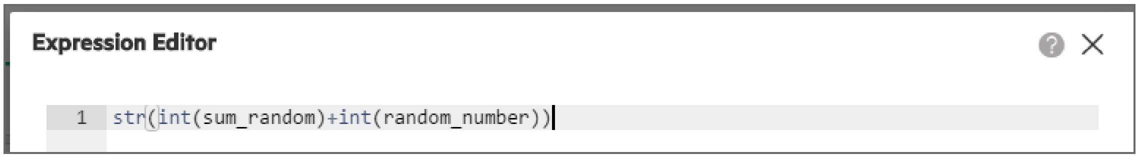
5. Define the aggregated output for the loop step as shown below.



Name	Value
sum_random	str(int(sum_random)+int(random_number))
random_number	random_number

> **Note:** The **random_number** output shown in the above image is optional.

The Python expression for the **sum_random** output using (aggregating) the outputs from the iterations is:

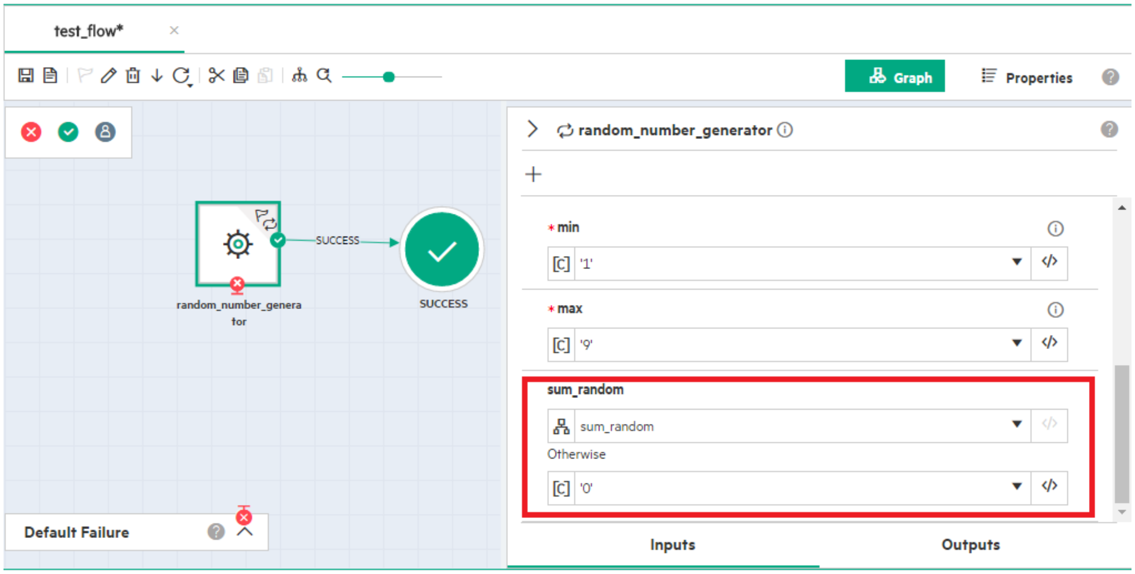


Expression Editor

```
str(int(sum_random)+int(random_number))
```

6. Add a new Step Input and rename it as the Output. For example the following image shows the Step Input name as **sum_random**. This **sum_random** step input is assigned from itself, or with **0**

when it has no value yet (at the first iteration).



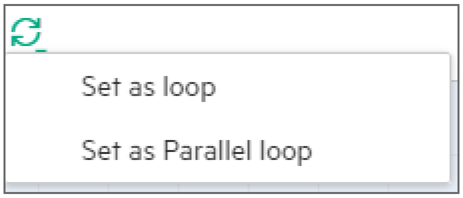
- min: [C] '1'
- max: [C] '9'
- sum_random: sum_random
- Otherwise: [C] '0'

7. Click **Save** 💾. The flow is saved.

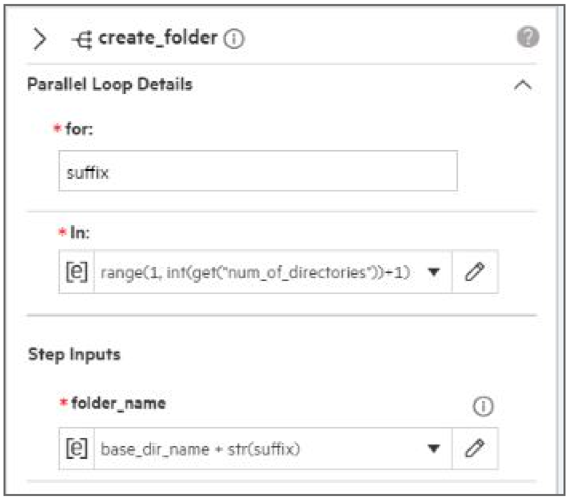
Create a flow with a parallel loop

In a parallel loop step, a new branch is created for each value in the collection and the branches run in parallel. The rest of the flow will continue only after all the branches have finished running.

1. Create a flow and select the step that you want to use as a parallel loop step.
2. Click the **Create Loop** button in the Authoring pane toolbar.
3. From the menu that appears, select **Set as a parallel loop**.



4. For a step that has been set as a loop, the **Step Properties** > **Inputs** tab has an extra **Loop Details** section, where you can define the collection that the step will iterate on.

   create_folder

   Parallel Loop Details

   * for:

   suffix

   * In:

   [e] range(1, int(get("num_of_directories"))-1)

   Step Inputs

   * folder_name

   [e] base_dir_name + str(suffix)

   - **For**: Enter the name of the variable that will be assigned the current value from the collection that is defined under **In**. This step is compulsory.

     This variable will be available while the loop step is running.

   - **In**: Define the collection to be iterated on. This step is compulsory.

     You can define the collection using a flow variable or system property. For example, you can create a system property with a comma delimited string (1,2,3,4,5) as the value and use this to define the collection.

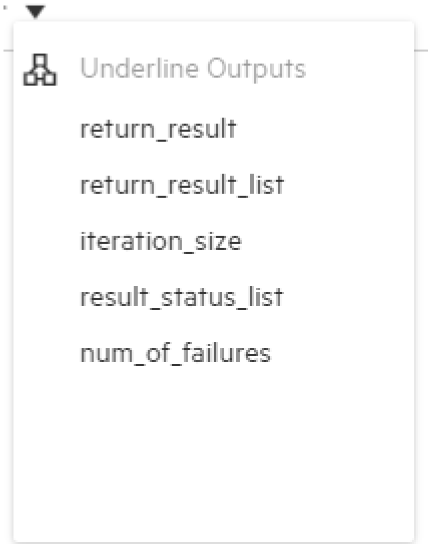
     You can also open the Expression Editor and write a Python expression that results in a list or map. For example:

     - An array: ['1','2','3','4'] or ['a','b','c','d']
     - A Python expression: map(str,range(1,5))
     - A map: {"key1":"value1","key2":"value2"}
     - A comma delimited string, which will be split automatically into a list: '1,2,3'.

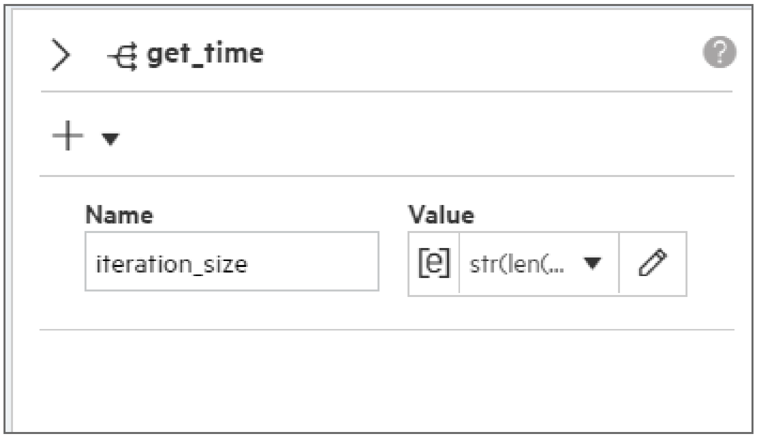
5. In the **Step Properties** > **Outputs** tab, define how the output will be published.

a. Click the **Add a step output** + button to add a new output to the step.

b. Choose one of the underline outputs to define the output:



c. Under **Value** for the output, open the Expression Editor and enter a Python expression to aggregate the outputs.



Name	Value
iteration_size	str(len(...

To define the step aggregation logic, you can use the **branches_context** construct. This is a list that includes the context of all the iterations. The **branches_context** is set according to the order of the executions, so that the execution that finishes first takes the first place in the list, and so on.

After all the branches of the parallel loop have completed, the execution of the flow continues with the output that is published. The expression of each output pair is evaluated and published to the flow's scope. The out-of-the-box outputs are **name** (first branch), **name_list**, **iteration_size**, **result_status_list**, and **num_of_failures**.

Examples:

- To retrieve the name variable from the first branch to finish:

  ```
  branches_context[0]['name']
  ```

- To publish aggregated output: (this iterates over the branches_context, which contains a list of all the iterations of the parallel flow, and takes the value of 'name' from each branch)

  ```
  str([str(x["name"]) for x in branches_context])
  ```

- To create a list of branch results: (as branches of a parallel flow complete, the branch results get placed into the **branches_context** list under the **branch_result** key)

  ```
  str([str(x["branch_result"]) for x in branches_context])
  ```

- To configure the iteration size.

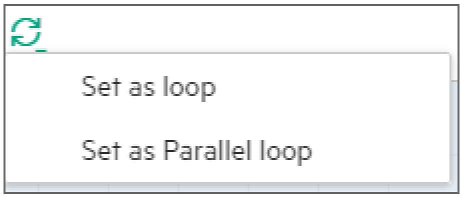
  ```
  str(len(branches_context))
  ```

- To configure the number of failures, use **num_of_failures**:

  ```
  str([x["branch_result"] for x in branches_context].count("FAILURE"))
  ```

Remove the loop from a step

1. Select the loop step.
2. Click the **Create Loop** button in the Authoring pane toolbar.
3. From the menu that appears, click the selected option to clear it.



View the CloudSlang Code

View the CloudSlang code about the inputs, outputs, and results of the step.

You can see information about the step's inputs, outputs, and navigation steps in the CloudSlang code for the flow.

To view the code, click the **Show as Text** button in the Authoring pane toolbar.. A text box appears, displaying the code. The details of the steps are displayed in the "workflow:" section. For details, see "YAML File Textual Representation" below.

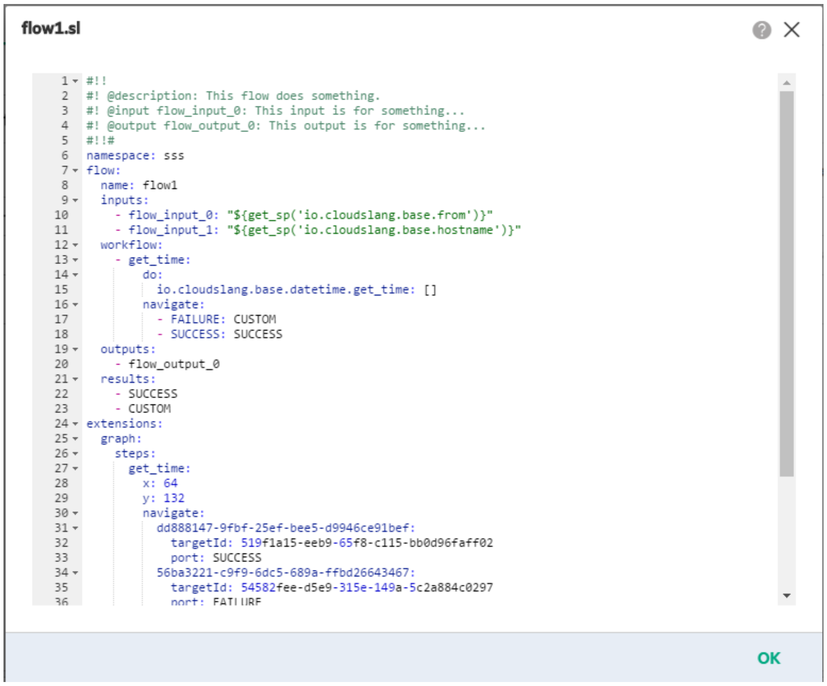
YAML File Textual Representation

You can view the YAML file corresponding to a selected flow, operation, decision, or system property, by clicking the **Show as Text** button in the Authoring pane toolbar.

The YAML file includes the textual representation of the UI editor, so that each action done in the UI editor, such as, for example, adding inputs, editing outputs, adding steps, is reflected in the YAML file.

> **Note:** It is not recommend to set a default constant value when an input and output is marked as sensitive. The values are displayed as asterisks in the file.

Example of a typical file:



```
#!!
#! @description: This flow does something.
#! @input flow_input_0: This input is for something...
#! @output flow_output_0: This output is for something...
#!!#
namespace: sss
flow:
  name: flow1
  inputs:
    - flow_input_0: "${get_sp('io.cloudslang.base.from')}"
    - flow_input_1: "${get_sp('io.cloudslang.base.hostname')}"
  workflow:
    - get_time:
        do:
          io.cloudslang.base.datetime.get_time: []
        navigate:
          - FAILURE: CUSTOM
          - SUCCESS: SUCCESS
  outputs:
    - flow_output_0
  results:
    - SUCCESS
    - CUSTOM
extensions:
  graph:
    steps:
      get_time:
        x: 64
        y: 132
        navigate:
          dd888147-9fbf-25ef-bee5-d9946ce91bef:
            targetId: 519f1a15-eeb9-65f8-c115-bb0d96faff02
            port: SUCCESS
          56ba3221-c9f9-6dc5-689a-ffbd26643467:
            targetId: 54582fee-d5e9-315e-149a-5c2a884c0297
            port: FAILURE
```

The first part (in green) displays the contents of the description of the item (flow, operation, decision, system property), as well as the description of input, output, and result.

```
#!!
#! @description: This flow does something.
#! @input flow_input_0: This input is for something...
#! @output flow_output_0: This output is for something...
#!!#
```

The second part (in blue), starting with **namespace:**, provides the CloudSland code that corresponds to the information about the selected flow or system property that were entered in the UI editor.

```
namespace: sss
flow:
  name: flow1
  inputs:
    - flow_input_0: "${get_sp('io.cloudslang.base.from')}"
    - flow_input_1: "${get_sp('io.cloudslang.base.hostname')}"
  workflow:
    - get_time:
        do:
          io.cloudslang.base.datetime.get_time: []
        navigate:
          - FAILURE: CUSTOM
          - SUCCESS: SUCCESS
  outputs:
    - flow_output_0
  results:
    - SUCCESS
    - CUSTOM
```

The third part (in blue), starting with **extensions:** provides the CloudSlang code that enables the graphical representation of the flow in the **Graph** tab. At run time, this section is ignored.

```
extensions:
  graph:
    steps:
      get_time:
        x: 64
        y: 132
        navigate:
          dd888147-9fbf-25ef-bee5-d9946ce91bef:
            targetId: 519f1a15-eeb9-65f8-c115-bb0d96faff02
            port: SUCCESS
          56ba3221-c9f9-6dc5-689a-ffbd26643467:
            targetId: 54582fee-d5e9-315e-149a-5c2a884c0297
            port: FAILURE
    results:
      SUCCESS:
        519f1a15-eeb9-65f8-c115-bb0d96faff02:
          x: 384
          y: 75
```

```
CUSTOM:
  54582fee-d5e9-315e-149a-5c2a884c0297:
    x: 360
    y: 230
```

Validate Flows

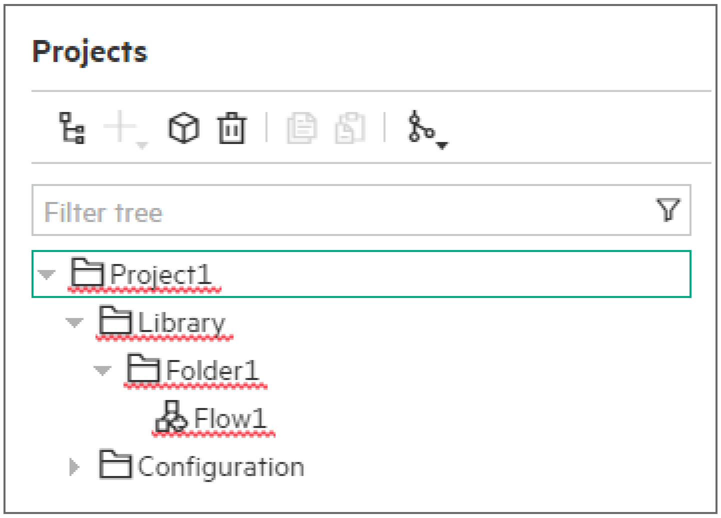
There are a number of built-in validation errors and warning in OO Designer.

It is highly recommended to check that your flows are valid before exporting your content to a content pack, as invalid content will **not** be exported to content packs.

This section provides the location of errors and warnings appearing in the UI.

Viewing Errors

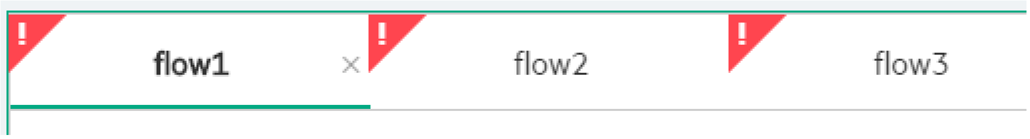
- **Projects or Dependencies pane.** If there is an error anywhere in the **Project** pane or **Dependencies** pane, (in a flow, operation, or system property), the item with the error is displayed with a red underline. All flows that use this item and the folders above it are also displayed with a red underline.

  When you fix the error, the red lines disappear under that item and under the folders that contain it or flows that use it.

As soon as you add or create a flow, OO Designer automatically runs the validation and displays all errors underlined. There is no need to do anything to start this process.

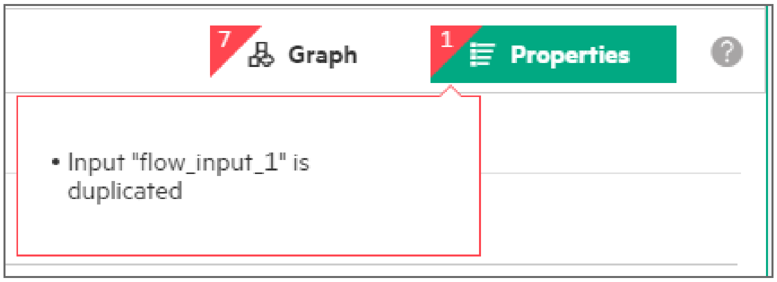
- **Authoring pane.**
  - **Flow tabs.** If there is any error on the flow\operation, an error flag appears on the tab. Click the error flag to display a tooltip with a detailed list of errors.

  - **Flow inner Graph or Properties tabs.**

    If a flow or its properties contain errors, an error flag is displayed in the **Graph** and/or **Properties** tabs at the top of the Authoring pane. This flag includes a number, representing the number of errors.

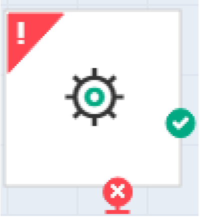
    Click the error flag to display a tooltip with a detailed list of errors.

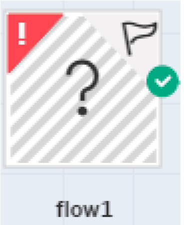
    - Input "flow_input_1" is duplicated

  - **Error indications on the step.** if there is an error in a step, that step appears with an error flag in the top left corner.

    > **Tip:** Click an error flag to display a tooltip with information about the error.

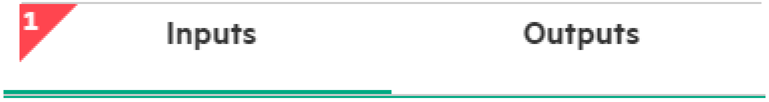
    

    If there is a step with a missing dependency (for example, a step that is based on a flow that was deleted), that step appears with a gray, striped background.



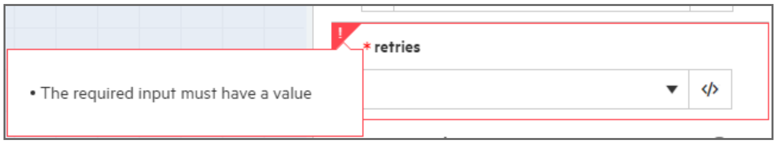
- **Errors on the step properties tabs.**

  If there is an error at the step inputs or outputs, a red error sign is displayed in the Step Properties pane, on the **Inputs** and/or **Outputs** tabs. This includes a number, representing the number of input or output errors in the step.

- **Errors inside step inputs tab and step outputs tab.**

  Errors in a specific input or output appear as a red error sign in the top left corner. Click to display the detailed error tooltip.

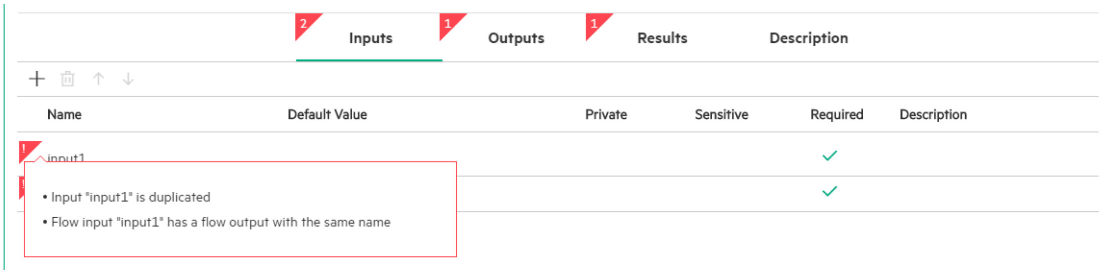
  • The required input must have a value

  While you are entering text in a field (for example, entering a flow input), if the value being entered will cause the underlying YAML file to be corrupted, OO Designer shows a red warning on the text field. If you focus out of the field at this point, the value will be reset to the previous value.

- **Error indication on the flow inputs, inputs, or results.**

  If there is an error at the flow inputs, outputs, or results, a red error sign is displayed in the **Inputs**, **Outputs**, or **Results** tabs in the flow **Properties** tab. This includes a number, representing the number of errors. Click the error flag to display the detailed error tooltip.

  Inside each tab, the elements with errors appear with a red error sign in the top left corner. Click the error flag to display the detailed error tooltip.

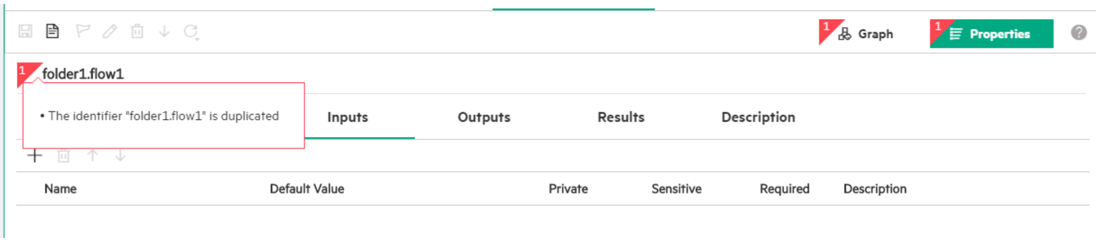


Name	Default Value	Private	Sensitive	Required	Description
input1				✓	
				✓	

- Input "input1" is duplicated
- Flow input "input1" has a flow output with the same name

- **Errors on flow identifier.**

  If the flow identifier already exists in the workspace, an error is issued and appears at the top left corner of the flow identifier. Click the error flag to display the detailed error tooltip.



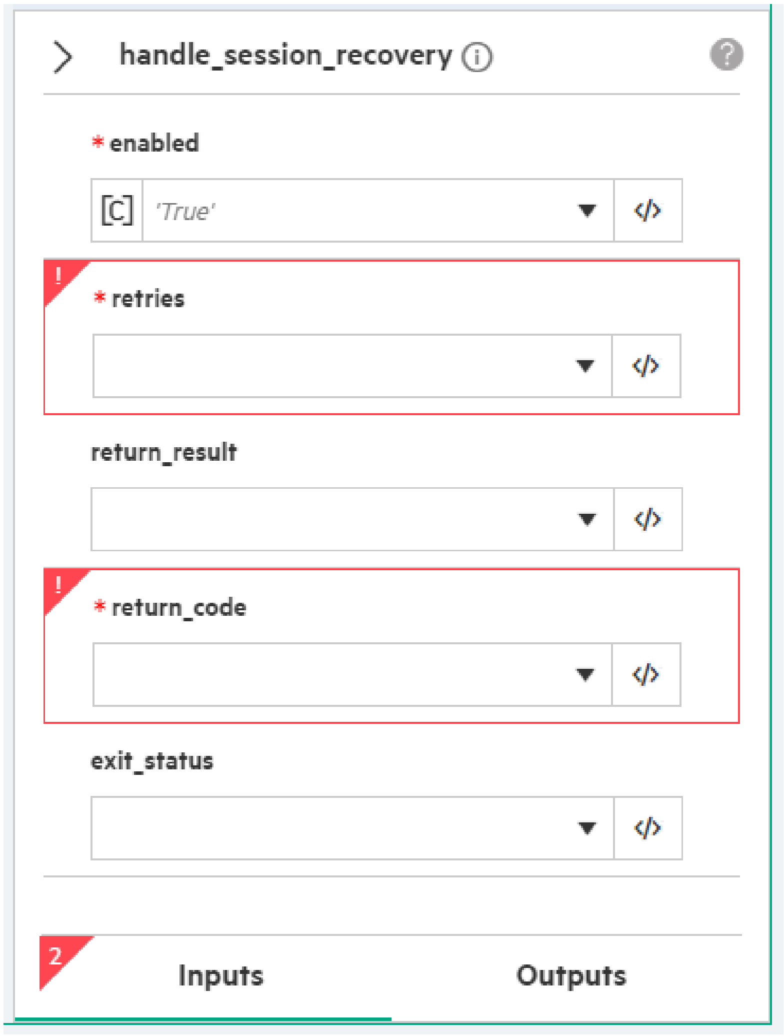
- The identifier "folder1.flow1" is duplicated

Name	Default Value	Private	Sensitive	Required	Description

Check for errors in the step properties inputs\outputs\results tabs

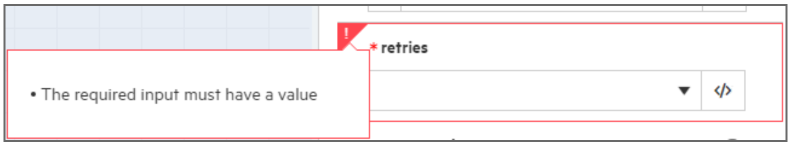
1. Click a step in the Authoring pane. The **Step Properties** pane appears at the right side of the canvas.

   The **Step Properties** pane contains the **Inputs**and **Outputs** tabs, with information about the selected step.



2. Expand or collapse this pane by clicking the **Expand** or **Collapse** button.

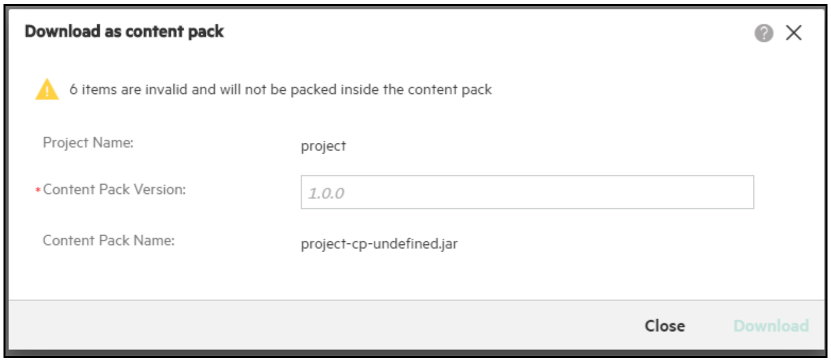
3. The tabs display any relevant warnings in the inputs, outputs, or results.



- The required input must have a value

Warnings

Warnings are issued if there is a problem, but it does not prevent the relevant operation to be performed. For example, if there is invalid content in a content pack that you try to download, the dialog box displays a warning message that there are invalid flows, which will not be packed inside the content pack.



Download as content pack

6 items are invalid and will not be packed inside the content pack

Project Name: project

*Content Pack Version: 1.0.0

Content Pack Name: project-cp-undefined.jar

Author Operations

An operation is an executable that performs a single action. It has metadata like a flow (inputs, outputs and results). For example, one operation checks a web page to see whether it contains specific text, and another operation copies a file.

More about operations

The action in an operation can be Java-based or Python-based.

There are two types of actions in CloudSlang:

- **Python-based actions.** See "Using Python Operations" on page 103.
- **Java-based actions.** See "Using Java Operations" on page 100.

Operations must be created externally . They can then be imported:

- As part of content packs, they are displayed in the **Dependencies** panel. They are read-only. They cannot be copied.
- Via Source Control Management (SCM), they are displayed in the **Projects** panel. They are read-only. They cannot be copied.

In OO Designer, you can use operations in flows by dragging the operation from the relevant content pack from the **Dependencies** pane. You can also import a repository that includes the operation by dragging it from the **Projects** panel. In both cases, the operation is read-only.

Operation File Structure

An operation file must end with the **.sl** extension and start with a namespace.

Example of an operation file with its namespace.

```
namespace: io.cloudslang.base.comparisons

operation:
  name: less_than_percentage
  inputs:
    - first_percentage
    - second_percentage
  python_action:
    script: |
      error_message = ""
      result = ""
      first_percentage_nr = first_percentage.replace("%", "")
      second_percentage_nr = second_percentage.replace("%", "")
      try:
          int_value1 = int(first_percentage_nr)
          int_value2 = int(second_percentage_nr)
          result = int_value1 < int_value2
      except ValueError:
          error_message = "Both inputs have to be integers"
  outputs:
    - error_message

  results:
    - LESS: ${error_message == "" and result}
    - MORE: ${error_message == "" and not result}
    - FAILURE
```

All CloudSlang files start with a namespace, which mirrors the folder structure in which the files are found. The namespace can be used by flows that call this operation.

Validating Operations

Operations are validated and will **not** work when:

- **Missing action.** Operations must contain an action.
- **Corrupted YAML.**
- **Duplicate inputs.** When an input is duplicated, or two inputs have the same name but use different cases (for example: Ping and ping).
- **Input\output\result name or value.**
  - Name cannot be empty
  - The combined input name, flow name, and folder path of the input (under the **Library** folder) must add up to no more than 255 characters.

    > **Note:** For localized characters, this limit is even lower. For example, in Japanese, the 255 characters need to be divided by 4, so the number of characters is only 64.

  - Name must be unique.

  It is recommended to manually validate the inputs that your operation will receive. For example, if you expect a number, make sure that the input is a string representing a number. If you expect an email address, check that the format complies with an email address format.

- **Missing rules in operation results.** Operation with more than one empty result (with no defined rule).

Operations can have multiple expression results, but can have only one mandatory empty default result, and it must be the last one.

- **Incorrect operation name.** When the name of the operation does not comply with the rules below:

  Name cannot be identical to the following reserved word: "on_failure".

  Names must be unique within their folders.

  Names are not case-sensitive, so names such as "Ping" and "ping" are considered duplicates. However, when calling an item, make sure to use the correct case.

  Names can contain alphanumeric characters, and underscores (_).

  Names cannot contain parentheses (()), square brackets ([]), curly braces ({}), hypens (-), spaces ( ), and dots (.).

  Names cannot be identical to the following Windows reserved words: **CON**, **PRN**, **AUX**, **CLOCK$**, **COM1**, **COM2**, **COM3**, **COM4**, **COM5**, **COM6**, **COM7**, **COM8**, **COM9**, **NUL**, **LPT1**, **LPT2**, **LPT3**, **LPT4**, **LPT5**, **LPT6**, **LPT7**, **LPT8**, **LPT9** .

Decisions

A Decision is a step that serves as a gateway for a decision in a flow. A Decision performs a calculation and then controls the direction of the flow, based on rules that are used to calculate the result. Decisions do not perform actual operations but only calculate where to navigate next. They are read-only. Their inputs, outputs and results can be viewed but not edited.

For details, see "Add Decisions to a Flow" on page 76.

Using Java Operations

A Java-based operation includes inputs, outputs, results, and one action.

Java-based actions are read only and you cannot add\copy\edit them. They can only be found under the Dependencies panel, or imported using Source Control Management (SCM) (you cannot edit and copy them even if they are located in the Projects panel). You can use them as steps in flows by dragging them from the **Dependencies** or the **Projects** pane.

Use the Operation Editor to display information about an operation.

Java Actions

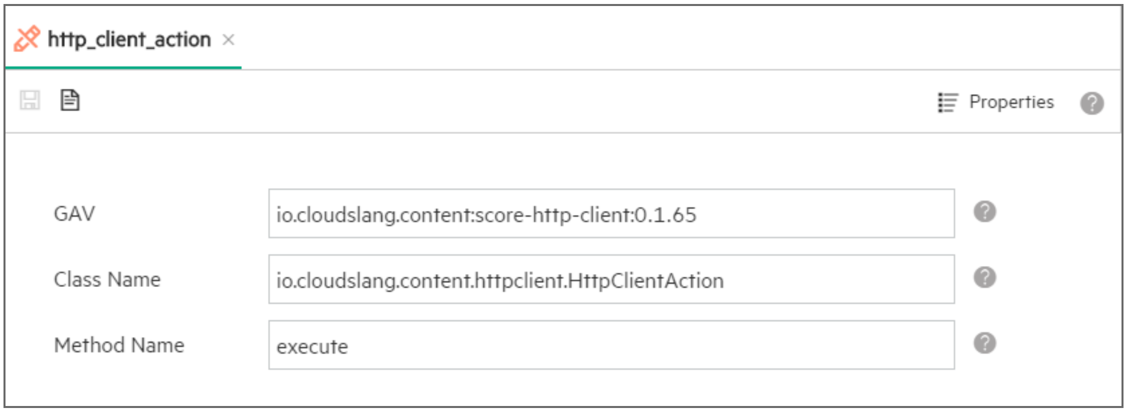
Java action contain the following elements:

- **GAV.** GAV is short for "Group, Artifact, Version". The GAV coordinate standard is the foundation for how Maven manages dependencies.
  - A group identifier groups a set of artifacts into a logical group
  - An artifact is an identifier for a software component
  - The version of a project follows the established convention of Major, Minor, and Point release versions.
- **Class Name.** This is the class within the GAV where the running method can be found.
- **Method Name.**This is the method to run within the action.

View the properties of a Java operation

1. Double-click an operation in the **Projects** or **Dependencies** pane to open it in the Operation Editor as read-only.

   The **Java Action** view displays the GAV (group, artifact, version), class name, and method name that define the Java action in the operation.

   GAV: io.cloudslang.content:score-http-client:0.1.65

   Class Name: io.cloudslang.content.httpclient.HttpClientAction

   Method Name: execute

2. Click the **Properties** Properties button to display the **Properties** view.

   You can return to the **Java Action** view, by clicking either the **Java Action** Java Action button or the **Back to Action** Back to action button.

3. Under the **Inputs** tab, view the inputs that the operation receives. Each input in an operation has a value that can be set in a number of ways. It may be a constant value, a python expression, the value of a flow variable or system property, and more.

   See "Inputs" on page 50.

4. Under the **Outputs** tab, view the values returned from the Java code. They will be sent back to the

calling step.

The Outputs tab lists any information that needs to be sent back to the calling step. The outputs are a list of key:value pairs where the key is the name of the output and the value is the expression to be returned, blank, constant, or more.

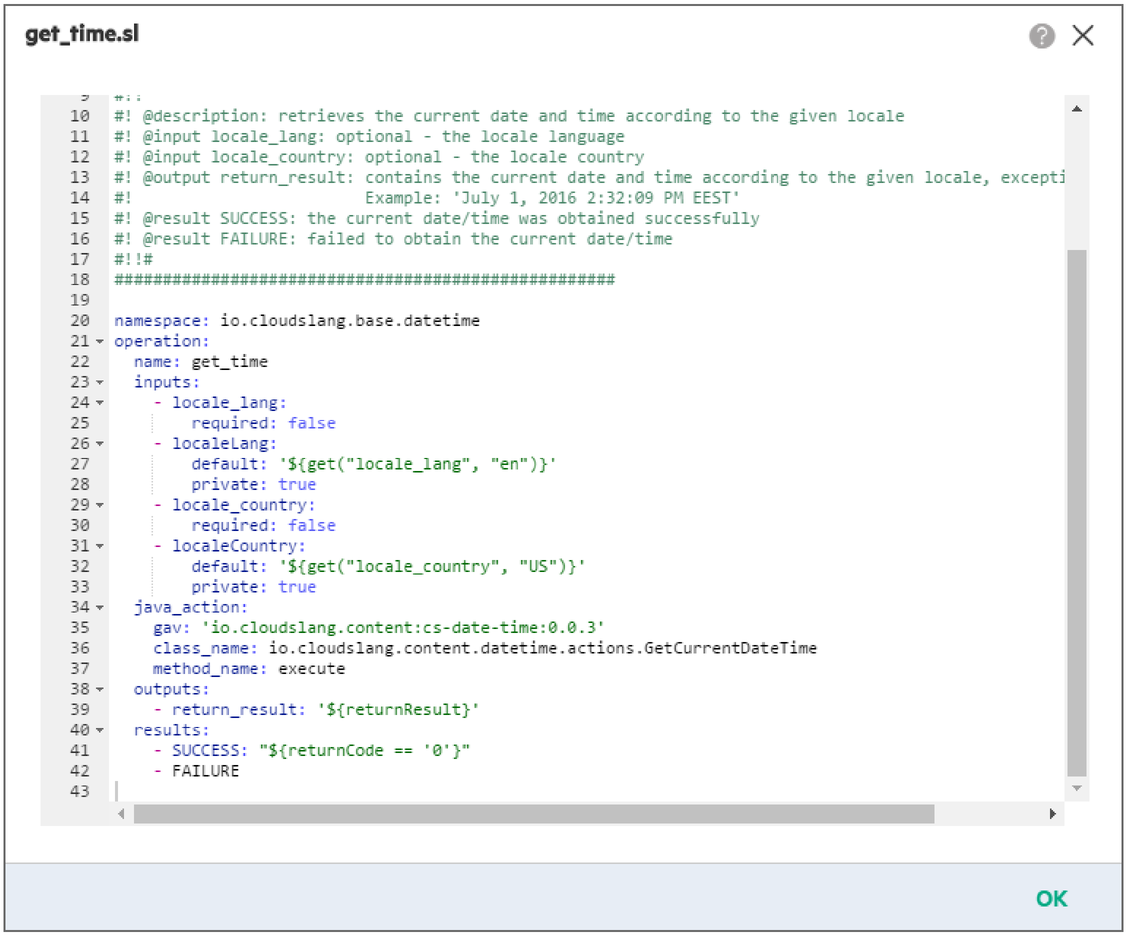
See "Outputs" on page 52.

5. Under the **Results** tab, view the results that will be returned to the calling flow.

   See "Results" on page 54.

6. Under the **Description** tab, view a description of what the operation does, if one was entered.

7. To view the code behind the operation, click the **Show as Text** 🗎 button. A text box appears, displaying the code.



```
#! @description: retrieves the current date and time according to the given locale
#! @input locale_lang: optional - the locale language
#! @input locale_country: optional - the locale country
#! @output return_result: contains the current date and time according to the given locale, excepti
#!                          Example: 'July 1, 2016 2:32:09 PM EEST'
#! @result SUCCESS: the current date/time was obtained successfully
#! @result FAILURE: failed to obtain the current date/time
#!!#
####################################################

namespace: io.cloudslang.base.datetime
operation:
  name: get_time
  inputs:
    - locale_lang:
        required: false
    - localeLang:
        default: '${get("locale_lang", "en")}'
        private: true
    - locale_country:
        required: false
    - localeCountry:
        default: '${get("locale_country", "US")}'
        private: true
  java_action:
    gav: 'io.cloudslang.content:cs-date-time:0.0.3'
    class_name: io.cloudslang.content.datetime.actions.GetCurrentDateTime
    method_name: execute
  outputs:
    - return_result: '${returnResult}'
  results:
    - SUCCESS: "${returnCode == '0'}"
    - FAILURE

```

8. Click **OK** to close the text box.

Using Python Operations

Python-based operations include inputs, outputs, results, and actions.

About python operations

Python operations can be copied from both the Projects panel and the Dependencies panel. You can import them. They are read-only.

Example:

```
namespace: io.cloudslang.base.comparisons

operation:
  name: less_than_percentage
  inputs:
    - first_percentage
    - second_percentage
  python_action:
    script: |
      error_message = ""
      result = ""
      first_percentage_nr = first_percentage.replace("%", "")
      second_percentage_nr = second_percentage.replace("%", "")
      try:
          int_value1 = int(first_percentage_nr)
          int_value2 = int(second_percentage_nr)
          result = int_value1 < int_value2
      except ValueError:
          error_message = "Both inputs have to be integers"
  outputs:
    - error_message

  results:
    - LESS: ${error_message == "" and result}
    - MORE: ${error_message == "" and not result}
    - FAILURE
```

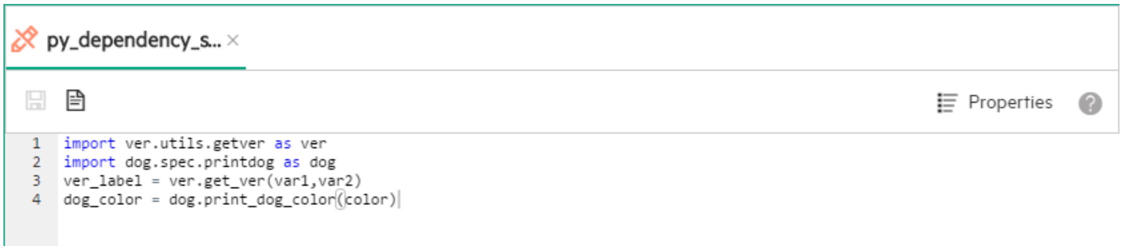
You can use operations as steps in flows by dragging them from the **Dependencies** pane or from the **Projects** pane.

Use the Operation Editor to display information about an operation.

View the properties of a Python operation

1. Double-click an operation in the **Projects** or the **Dependencies** pane to open it in the Operation Editor as read-only.

   The **Python Action** view displays the python action script.

   ```
   import ver.utils.getver as ver
   import dog.spec.printdog as dog
   ver_label = ver.get_ver(var1,var2)
   dog_color = dog.print_dog_color(color)
   ```

2. Click the **Properties** ☰ Properties button to display the **Properties** view.

   You can return to the **Python Action** view, by clicking either the **Python Action** ⚙ Python Action button or the **Back to Action** Back to action ↩ button.

3. Under the **Inputs** tab, view the inputs that the operation takes. Each input in an operation has a value that can be set in a number of ways. It may be a constant value, a python expression, the value of a flow variable or system property, and more.

   See "Inputs" on page 50.

   a. Under the **Outputs** tab, view the values returned from the python script, which will be sent back to the calling step.

      The outputs are a list of key:value pairs where the key is the name of the output and the value is the expression to be returned, blank, constant, or more.

      For example:

      ```
      ‘outputs:
          - available: ${vacant}’
      ```

      will return the value of a variable called ‘vacant’ in the python code, with the name of ‘available’ to the calling flow. More complicated expressions can be passed, such as trimming, parsing etc. Output expressions must evaluate to strings.

      If the value is blank, the output result will be the name of the output, which will translate to a python variable that is declared inside a python code. For example, ‘outputs: -vacant’ will return the value of the variable vacant under its’ original name as declared inside the python code.

      See "Outputs" on page 52.

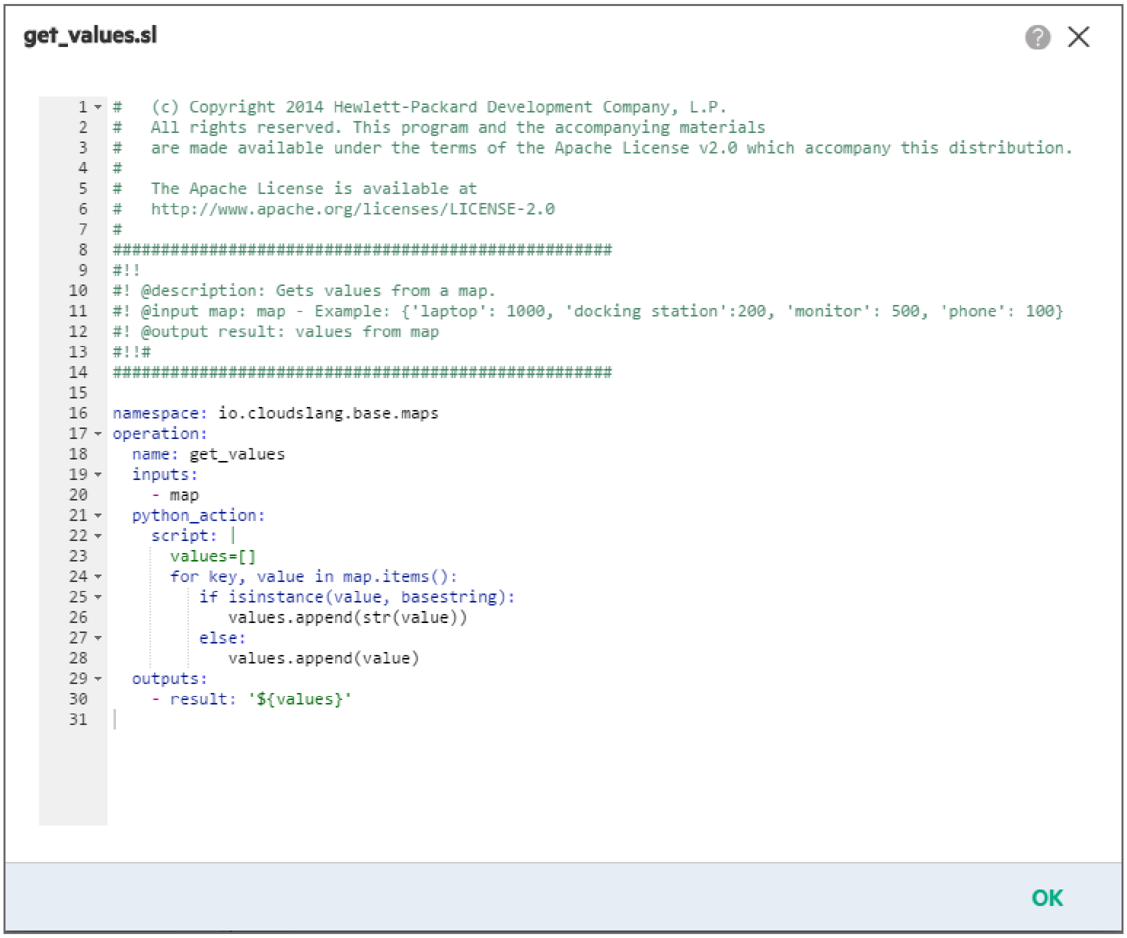
4. Under the **Results** tab, view the results that will be returned to the calling step.

   See "Results" on page 54.

5. Under the **Description** tab, view a description of what the operation does, if one was entered.

6. To view the code behind the operation, click the **Show as Text** 🗎 button. A text box appears, displaying the code.

   > **Note:** Python scripts are not validated inside OO Designer.

   get_values.sl

   ```
   #   (c) Copyright 2014 Hewlett-Packard Development Company, L.P.
   #   All rights reserved. This program and the accompanying materials
   #   are made available under the terms of the Apache License v2.0 which accompany this distribution.
   #
   #   The Apache License is available at
   #   http://www.apache.org/licenses/LICENSE-2.0
   #
   ####################################################
   #!!
   #! @description: Gets values from a map.
   #! @input map: map - Example: {'laptop': 1000, 'docking station':200, 'monitor': 500, 'phone': 100}
   #! @output result: values from map
   #!!#
   ####################################################

   namespace: io.cloudslang.base.maps
   operation:
     name: get_values
     inputs:
       - map
     python_action:
       script: |
         values=[]
         for key, value in map.items():
            if isinstance(value, basestring):
               values.append(str(value))
            else:
               values.append(value)
     outputs:
       - result: '${values}'

   ```

7. Click **OK** to close the text box.

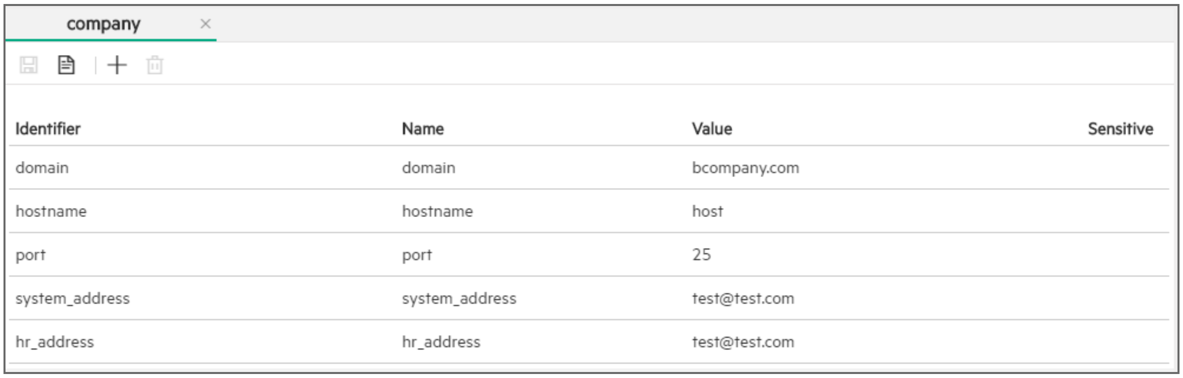
Author System Properties

System Properties are global variables with values that do not change. Once created, you can use these system properties in any flow without having to recreate a flow variable each time. The flow referenced to a system property obtains the value of the system property.

More about system properties

> For example, you can set a system property whose value is the URL of a server, and then use this property in a flow that runs a series of actions on the server. Rather than entering the URL each time, you can just select the property.

System properties are stored in system properties files located in the **Configuration/System Properties** folder.



Identifier	Name	Value	Sensitive
domain	domain	bcompany.com	
hostname	hostname	host	
port	port	25	
system_address	system_address	test@test.com	
hr_address	hr_address	test@test.com	

(Optional) You can add extra folders, to organize the system properties files.

Naming System Property Files

- The name of a system property is limited to 128 characters.
- Valid characters for system property names are: a-z A-Z, 0-9, underscore (_), and hyphen (-).

  Period (.) can be used as delimiter (only in the middle of the name).
- The combined name and path of a system property (under the **Configuration/System properties** folder) is limited to 220 characters.
- Names must be unique within their folders.
- Names are not case-sensitive, so names such as "Ping" and "ping" are considered duplicates. However, when calling an item, make sure to use the correct case.
- A system property name cannot include spaces.
- Names cannot be identical to the following Windows reserved words: **CON**, **PRN**, **AUX**, **CLOCK$**, **COM1**, **COM2**, **COM3**, **COM4**, **COM5**, **COM6**, **COM7**, **COM8**, **COM9**, **NUL**, **LPT1**, **LPT2**, **LPT3**, **LPT4**, **LPT5**, **LPT6**, **LPT7**, **LPT8**, **LPT9** .
- If you enter a name with an error, an error indication is displayed. When you focus out of the field, the value will be reset and a default name assigned.

In certain cases, if the change is causing the YAML to be corrupted (for example name too long) the name will be reset. Otherwise, it will not be reset and an error indication is displayed.

Create a system properties file containing system properties

1. In the **Projects** pane, expand the **Configuration** folder.
2. From the **System Properties** folder or a subfolder within the **System Properties** folder, click the **New** +▾ button and select **System Properties File**.

   > **Note:** You can also click the **New** +▾ button to create folders in which to store the system properties files.

3. Enter a name for the **system properties file** and click **OK**.
4. In the **Authoring** pane, under the **system properties file** tab, click the **New** + button to add a new system property to the system properties file.
5. Enter the details of the **system property** in the columns:
   - **Identifier**: Generated automatically, based on the namespace and the name that you provide in the **Name** column.
   - **Name**: Enter a unique name.
   - **Value**: Enter a value for the system property.
   - **Sensitive**: Select the check box, if the system property contains sensitive data. During runtime, the value will be encrypted.

     > **Tip:** It is recommend not to set a value when a system property is marked as sensitive. The sensitive values appear as asterisks in the file.

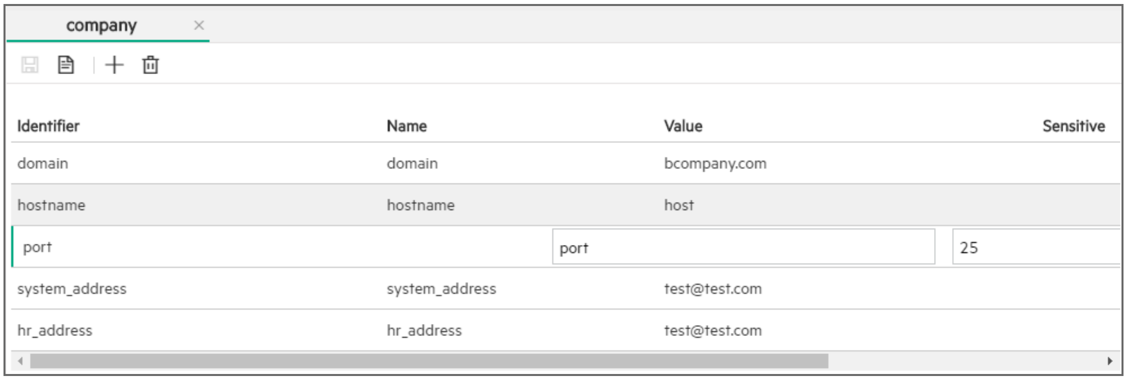
     As you enter text in a field, if the value being entered will cause the underlying YAML file to be corrupted, OO Designer shows a warning on the text field. If you select a different field at this point, the value will be reset to the previous value.
6. Repeat to add more system properties to the system properties file.
7. Click the **Save** 💾 button in the **Authoring** pane.
8. Click **Show as Text** 🗎 to display the system properties file information in YAML code and click **OK** to close.

Manage individual system properties

1. To edit a System Property:

   a. Open a system properties file containing it, and then double-click the cell you want to edit.

   > You can use the **Tab** key to move between cells in a row.



Identifier	Name	Value	Sensitive
domain	domain	bcompany.com	
hostname	hostname	host	
port	port	25	
system_address	system_address	test@test.com	
hr_address	hr_address	test@test.com	

   b. After making changes, click the **Save** button in the **Authoring** pane.

2. To delete a **System Property** from within a system properties file:

   a. Click on the system property row to select.

   b. Click the **Delete** button.

   c. Click the **Save** button in the **Authoring** pane.

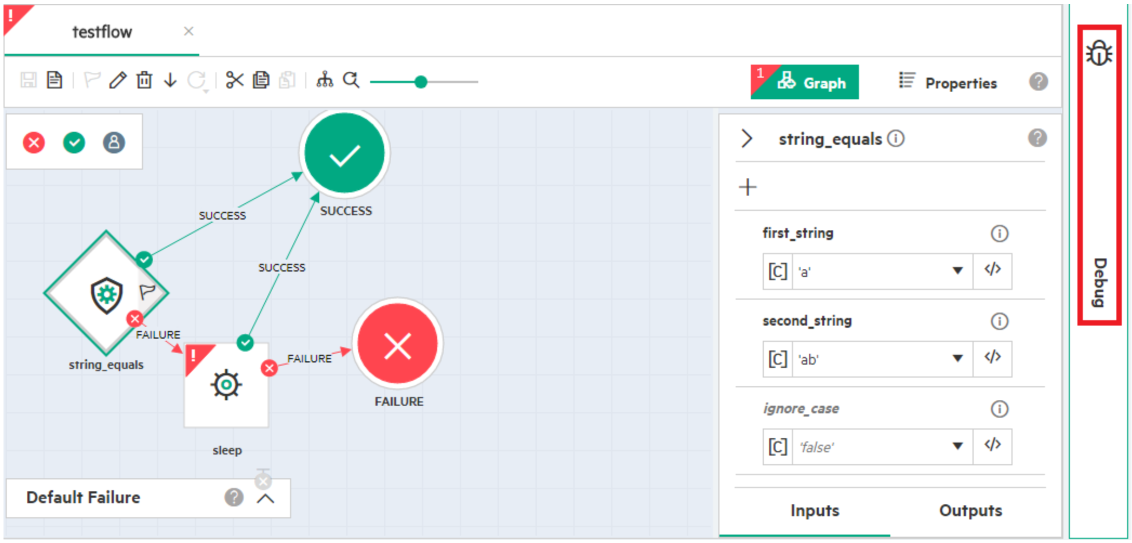
   > **Note:** If you delete a system property that is used in a flow, this system property will still be shown as selected in the flow. You need to replace this in the flow.

Debug the Flow

To debug the a Flow, perform the following steps:

1. Open an existing Flow from the project pane or from an imported Content Pack.

2. Click 🐞 or inverted **Debug** text to expand the Debugger pane.



string_equals

first_string [C] 'a'

second_string [C] 'ab'

ignore_case [C] 'false'

3. Click **Start** ▷ or press **Alt + F11** in the keyboard to begin the debug process.

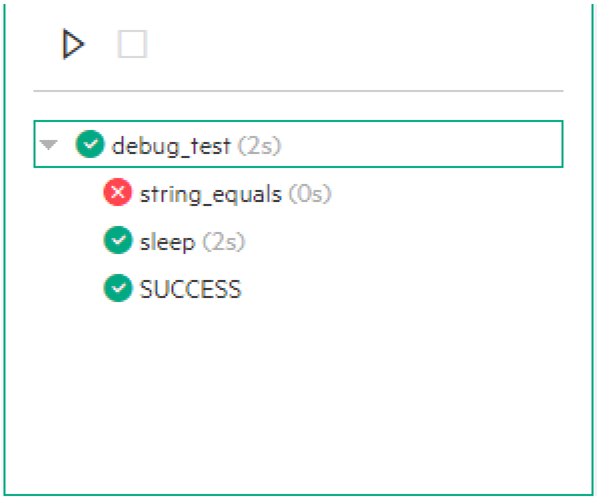
> **Note:** The start debug icon is disabled for an invalid flow. A relevant message is also displayed when pointing the cursor on the Debug icon.

> **Note:** The debugging of Operations, Decisions, and System Properties are not enabled in this version.

> In the multi-author environment:
> - A user can debug multiple flows at the same time.
> - A user cannot debug the same flow while it is still in the debugging process.
> - Multiple users connected to one Designer can debug the same flow in their work-spaces.

4. Click **Cancel** □ or press **Alt + F12**, if you want to cancel the debug process.

5. After debugging the flow, you can view the outputs and results of the flow in a **Run Tree** pane as displayed in the following example image.



debug_test (2s)

string_equals (0s)

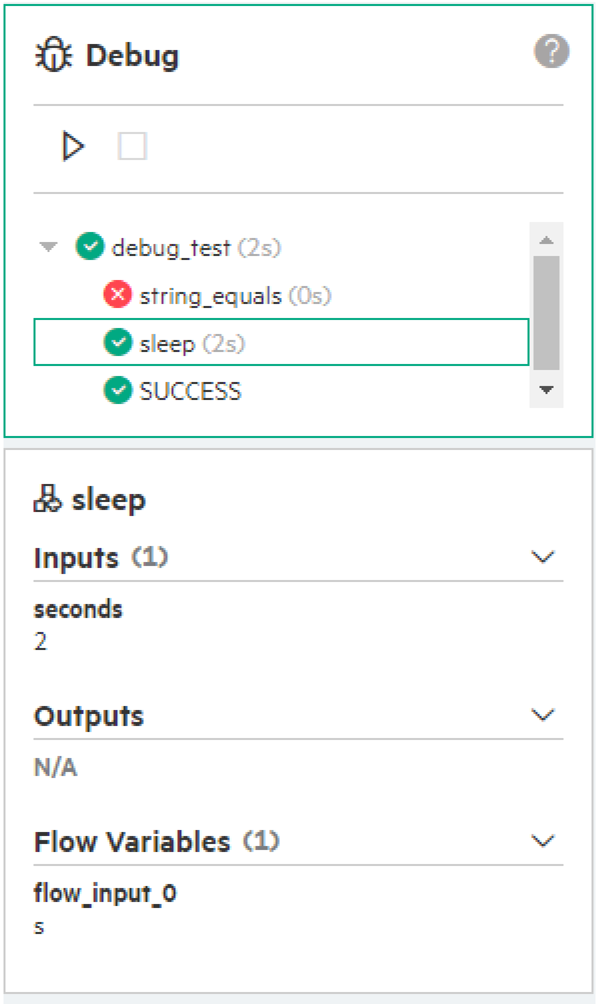
sleep (2s)

SUCCESS

The following icons are displayed in the Run Tree pane to indicate the status of a debug process:

Icon	Description
⟳	Flow is currently in debugging process
✖	Failure result
✓	Success result
⊖	Debug canceled or terminated abruptly
◆	Runtime exception
🔒	Custom result

6. Select any node in the **Run Tree** to check the context of that step in the **Context Inspector**. The following image displays the *input*, *output*, and *flow variables* for *sleep* node.



Inputs (1)

seconds
2

Outputs

N/A

Flow Variables (1)

flow_input_0
s

Create and Deploy a Content Pack

When you have finished authoring your content, you can export it as a content pack.

> **What is a content pack?** A content pack is the outcome of your project. It is collection of valid flows, decisions, operations, and system properties. It is packaged as a .jar file that is ready to be deployed in HPE OO Central or imported as a dependency into OO Designer.

Get the project ready to be exported to a content pack

1. Make sure that you have saved all changes in the workspace.
2. Check that the content in the project is valid and that there are no errors.

   Before exporting your content to a content pack, it is strongly recommended to check that your flows are valid. If the project contains invalid flows, decisions, operations, or system properties, a warning message is displayed, listing the number of invalid items that will be excluded from the content pack. If this occurs, it is recommended to fix the errors and then to try again as invalid content will **not** be exported. For details, see "Validate Flows" on page 92.

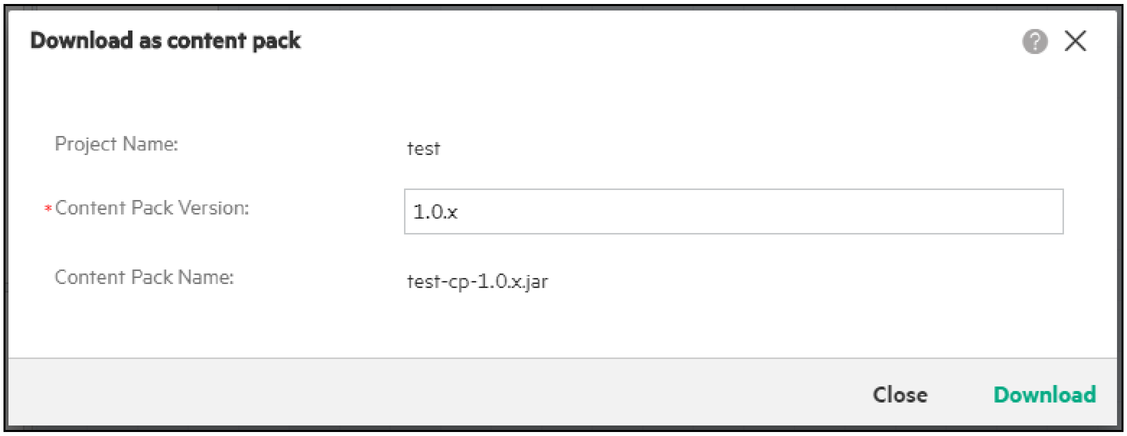
Download a project as a content pack

1. In the **Projects** pane, select the project from which you want to create a content pack.
2. Click the arrow next to the **Create Content Pack** button and select **Download as content pack**.

   > **Note:** The **Create Content Pack** button is only enabled when a project is selected.

   If there are unsaved editors open, the Save Changes dialog box gives you the option to save the changes. Click **Save All** to save all changes in the open editors.
3. OO Designer validates the project before creating a content pack. If the project contains invalid flows, operations, or system properties, these are not included.
4. In the dialog box, enter the content pack version number.

   > **Note:** Version numbers can be a maximum of 50 characters long. Make sure that the version number does not include invalid characters, such as: \/:*?"<>|
5. Click **Download**.
   The content pack is downloaded locally.



Download as content pack

Project Name: test

*Content Pack Version: 1.0.x

Content Pack Name: test-cp-1.0.x.jar

If there is invalid content in the project, the dialog box displays a warning that there are invalid items, which will not be packed inside the content pack.

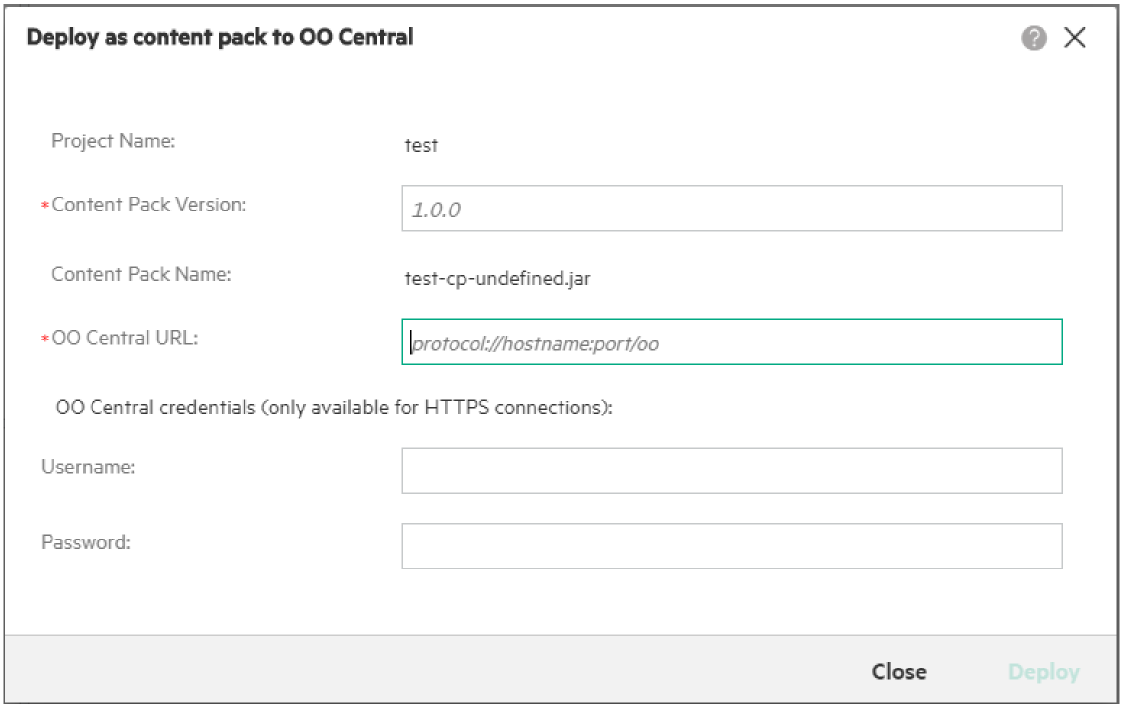
Create a content pack and deploy it to OO Central

> **Note:** To do this you must have the Manage Content permission in OO Central.

1. In the **Projects** pane, select the project from which you want to create and deploy a content pack.
2. Click the arrow next to the **Create Content Pack** ⬡ button and select **Deploy as content pack to OO Central**.

   > **Note:** The **Create Content Pack** ⬡ button is only enabled when a project is selected.

   If there are unsaved editors open, the Save Changes dialog box gives you the option to save the changes. Click **Save All** to save all changes in the open editors.
3. OO Designer validates the project before creating a content pack. If the project contains invalid flows, decisions, operations, or system properties, these are not included.
4. In the dialog box, enter the content pack version number.



Deploy as content pack to OO Central

Project Name: test

*Content Pack Version: 1.0.0

Content Pack Name: test-cp-undefined.jar

*OO Central URL: protocol://hostname:port/oo

OO Central credentials (only available for HTTPS connections):

Username:

Password:

> **Note:** Version numbers can be a maximum of 50 characters long. Make sure that the version number does not include invalid characters, such as: \/:*?"<>|

5. Enter the URL of OO Central.

   ◦ If the URL uses the HTTP protocol, a warning message displays that HTTPS is recommended, for security reasons. The OO Central **Username** and **Password** credential fields are disabled. However, you are still able to deploy the content pack. The validation only checks the format of the URL, not its existence. The format should be: **<protocol>://<hostname>:<port>/oo**.

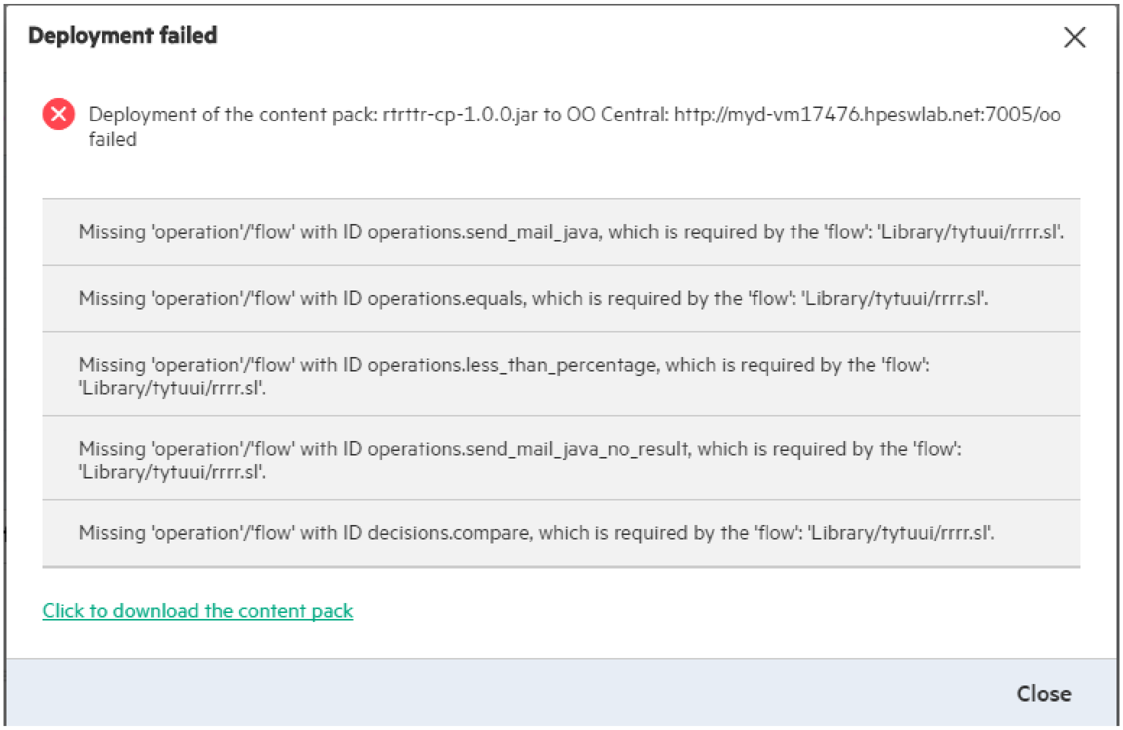
   ◦ If the URL uses the recommended HTTPS protocol, the OO Central **Username** and **Password** credential fields are enabled for inputs.

6. Enter the OO Central **Username** and **Password** credentials, and then Click **Deploy**.

   If there is invalid content in the project, the dialog box displays a warning that there are invalid items, which will not be packed inside the content pack.

   If the operation fails before the deployment to Central, for example, during the creation of the content pack, an error is displayed in the dialog box.

   If the operation fails during the deployment in Central, the list of problematic items is shown in the dialog box.



Deployment failed

Deployment of the content pack: rtrttr-cp-1.0.0.jar to OO Central: http://myd-vm17476.hpeswlab.net:7005/oo failed

Missing 'operation'/'flow' with ID operations.send_mail_java, which is required by the 'flow': 'Library/tytuui/rrrr.sl'.

Missing 'operation'/'flow' with ID operations.equals, which is required by the 'flow': 'Library/tytuui/rrrr.sl'.

Missing 'operation'/'flow' with ID operations.less_than_percentage, which is required by the 'flow': 'Library/tytuui/rrrr.sl'.

Missing 'operation'/'flow' with ID operations.send_mail_java_no_result, which is required by the 'flow': 'Library/tytuui/rrrr.sl'.

Missing 'operation'/'flow' with ID decisions.compare, which is required by the 'flow': 'Library/tytuui/rrrr.sl'.

You can still download the content pack locally by clicking **Click to download the content pack**.

7. If the content pack is deployed successfully to OO Central, you can also click **Click to download the content pack**.

> **Note:** If you are deploying to OO Central using a HTTPS URL, and the Central server certificate is not trusted, a dialog box opens showing the certificate details. The deployment process starts only if you approve the certificate.

How to use CloudSlang files created outside OO Designer in OO Designer

Before importing CloudSlang content to OO Designer, you need to create a content pack.

You can also create content packs with OO Shell for Authoring (OOSHA) and use it in OO Designer as a dependency if it is has a CloudSlang content. After running OOSHA, you import the content pack to OO Designer; the content pack then appears in the Dependencies section on the lower left. For details, see the relevant documentation.

> **Note:** Before creating the content pack, it is recommended to run the build tool in CloudSlang. The build tool validates the content so it doesn't fail during import to OO Designer.

1. Create a folder with the name of your content pack, and create the folder structure inside it, as follows:

   <Content Pack folder name>\Content\Library

   <Content Pack folder name>\Lib

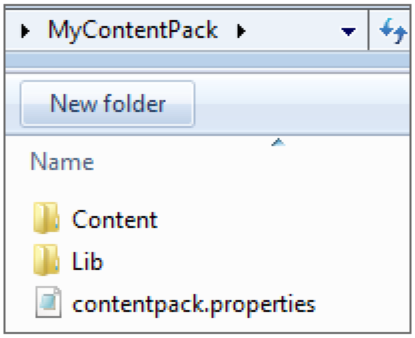
2. Place your CloudSlang content in the **Library** folder. You can organize it into sub-folders inside the **Library** folder.

3. Create a text file and name it **contentpack.properties**.

4. Add the following properties to the file:

   ```
   content.pack.name=slang-content-master
   content.pack.version=1.0.0
   content.pack.description=HP OO slang-content-master
   content.pack.publisher=OO
   ```

5. Store the **contentpack.properties** file at the top level of the content pack folder, at same level as the **Content** and **Lib** folders.



6. Create a .jar file from the content pack folder.

   > **Note:** You can do this using any tool that creates zip files. Create the file as a **zip** file and rename the suffix to **jar**.

7. Continue with the steps in Import content packs to the workspace.

Keyboard Shortcuts for OO Designer

To enhance productivity and accelerate navigation and common repetitive actions, OO Designer offers keyboard shortcuts as alternatives to clicking buttons and selecting options in the UI.

The main operation key for shortcuts is Alt, together with one letter.

> **Tip:** Print this section for easy access to the keyboard shortcuts.

Global	
Click OK	`Enter`
Click Cancel	`Esc`
Projects Pane	
Open the selected tree node in the Authoring pane	`Enter`
Copy the selected tree node	`Ctrl + C`
Paste the selected tree node	`Ctrl + V`
Delete the selected tree node	`Delete`
Create a new project	`Alt + P`
Create a new folder	`Alt + L`
Create a new flow	`Alt + W`
Create a new system property	`Alt + S`
Create a content pack	`Alt + C`
Dependencies Pane	
Import a content pack	`Alt + I`
Graph Pane	
Copy a selected step	`Ctrl + C`
Cut a selected step	`Ctrl + X`
Paste a selected step	`Ctrl + V`
Debug Pane	
Start debug	`Alt + F11`

Global	
Cancel debug	`Alt + F12`

Send documentation feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Use (Operations Orchestration Designer 1.2)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to oo_ie@hpe.com.

We appreciate your feedback!

Hewlett Packard
Enterprise

Java™
COMPATIBLE