Hitachi Ops Center Automator

11.0.4

Service Builder User Guide

This manual describes how to use Ops Center Automator Service Builder. Service Builder enables you to create and manage the service templates and associated plug-ins for automating data center tasks.

MK-99AUT002-21
April 2025

Contents

Preface

This manual describes how to use Ops Center Automator Service Builder.

Product version

This document revision applies to Hitachi Ops Center Automator v11.0.4-00 or later.

Release notes

Read the release notes before installing and using this product. They may contain requirements or restrictions that are not fully described in this document or updates or corrections to this document. Release notes are available on the Hitachi Vantara documentation website: https://docs.hitachivantara.com.

Conventions for storage capacity values

Physical storage capacity values (for example, disk drive capacity) are calculated based on the following values:

Physical capacity unit	Value
1 kilobyte (KB)	1,000 (10^3) bytes
1 megabyte (MB)	1,000 KB or $1{,}000^2$ bytes
1 gigabyte (GB)	1,000 MB or $1{,}000^3$ bytes
1 terabyte (TB)	1,000 GB or $1{,}000^4$ bytes
1 petabyte (PB)	1,000 TB or $1{,}000^5$ bytes
1 exabyte (EB)	1,000 PB or $1{,}000^6$ bytes

Logical capacity values (for example, logical device capacity, cache memory capacity) are calculated based on the following values:

Logical capacity unit	Value
1 block	512 bytes
1 cylinder	Mainframe: 870 KB Open-systems: ▪ OPEN-V: 960 KB ▪ Others: 720 KB
1 KB	1,024 (2^{10}) bytes
1 MB	1,024 KB or $1{,}024^2$ bytes
1 GB	1,024 MB or $1{,}024^3$ bytes
1 TB	1,024 GB or $1{,}024^4$ bytes
1 PB	1,024 TB or $1{,}024^5$ bytes
1 EB	1,024 PB or $1{,}024^6$ bytes

Accessing product documentation

Product user documentation is available on: https://docs.hitachivantara.com. Check this site for the most current documentation, including important updates that may have been made after the release of the product.

Getting help

The Hitachi Vantara Support Website is the destination for technical support of products and solutions sold by Hitachi Vantara. To contact technical support, log on to the Hitachi Vantara Support Website for contact information: https://support.hitachivantara.com/en_us/contact-us.html.

Hitachi Vantara Community is a global online community for Hitachi Vantara customers, partners, independent software vendors, employees, and prospects. It is the destination to get answers, discover insights, and make connections. **Join the conversation today!** Go to community.hitachivantara.com, register, and complete your profile.

Comments

Please send comments to doc.feedback@hitachivantara.com. Include the document title and number, including the revision level (for example, -07), and refer to specific sections and paragraphs whenever possible. All comments become the property of Hitachi Vantara LLC.

Thank you!

Chapter 1: Overview of Automator Service Builder

Service Builder has a powerful interface for managing and creating service templates that automate tasks, supply operating parameters for provisioning and allocating storage resources, and automate IT processes for your data center.

The service templates are based on plug-ins that serve as the building blocks for running scripts, issuing commands on a system, and supplying the values for the input and output properties and variables used when tasks run. The service templates and plug-ins can be linked together as a sequence of steps that dictate the flow of processes for a set of tasks in the main service template.

Because Service Builder is integrated with Ops Center Automator, there is no need to install Ops Center Automator on a separate test environment, and files and folders remain in a central location. Service Builder also enables you to test services, plug-ins, configuration files, and mapping parameters before they are released so that quality and functionality are ensured.

You can use the preconfigured collection of standard service templates and associated plug-ins, or minimally modify them, to perform many of the more common data center tasks. If the standard service templates and plug-ins are not adequate for your site, you can build your own service templates and plug-ins and define the types of tasks to run. You can also specify how to configure the operating parameters, user information, and connection settings and customize elements of the user interface required when submitting a service.

After you prepare the template with the required plug-ins and define the settings for the input and output properties, Service Builder guides you through the final stages of running, debugging, and releasing the service templates for use in your data center.

About Service Builder

Service Builder enables you to create and manage the service templates and associated plug-ins for automating data center tasks.

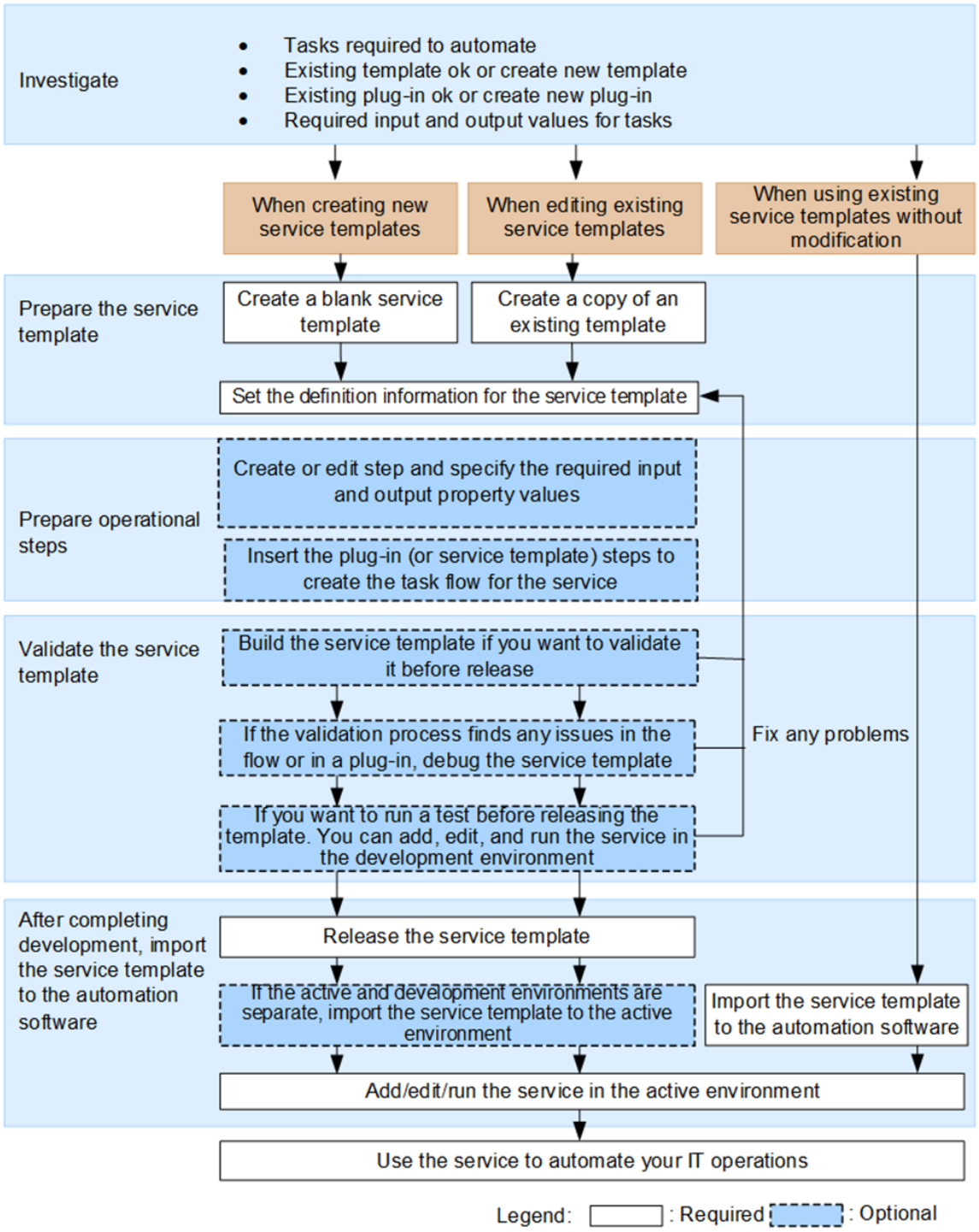
When working with service templates, you have the option of using one of the standard service templates, modifying an existing template to meet your needs, or creating a new template. Service templates comprise a sequence of plug-ins that run scripts, run commands, and provide required operational parameters to accomplish various tasks.

Service Builder can create a flow with plug-ins, and package these elements into a customized service template package. Service templates and plug-ins, both standard and customized, can be reused, saving you development time and effort.

Service Builder enables you to:

- Easily manage (copy, edit, delete, update and so on) service templates and plug-ins through a convenient menu-driven interface.
- Use or customize the standard service templates and plug-ins or create new ones to run the required scripts and the commands to complete the tasks for your site.
- Establish the flow of steps in which a series of plug-ins are run according to their placement by simply drawing connector lines between the components.
- Specify the values for the input and output properties associated with a plug-in that provide the host details, user information, and connections settings that are required to run a service, handle errors, and generate alerts.
- Run commands, run scripts, and make use of variables required when running service.
- Categorize tasks by user group or service classification.
- Customize the UI by controlling the type of icons and graphics that appear, the types of instructions and the information a user sees when supplying the details for submitting a service.
- Debug your service templates and eventually release them for deployment in your data center.
- Import and export service templates and settings so you can reuse them.
- Import and export property values that are stored in a file so that the setting and values for a service are supplied quickly and consistently from one session to the next.

The following shows the general workflow when working with service templates and their associated plug-ins.



Use this guide to design and create service templates, including the plug-ins that create the steps in a service template, with Service Builder.

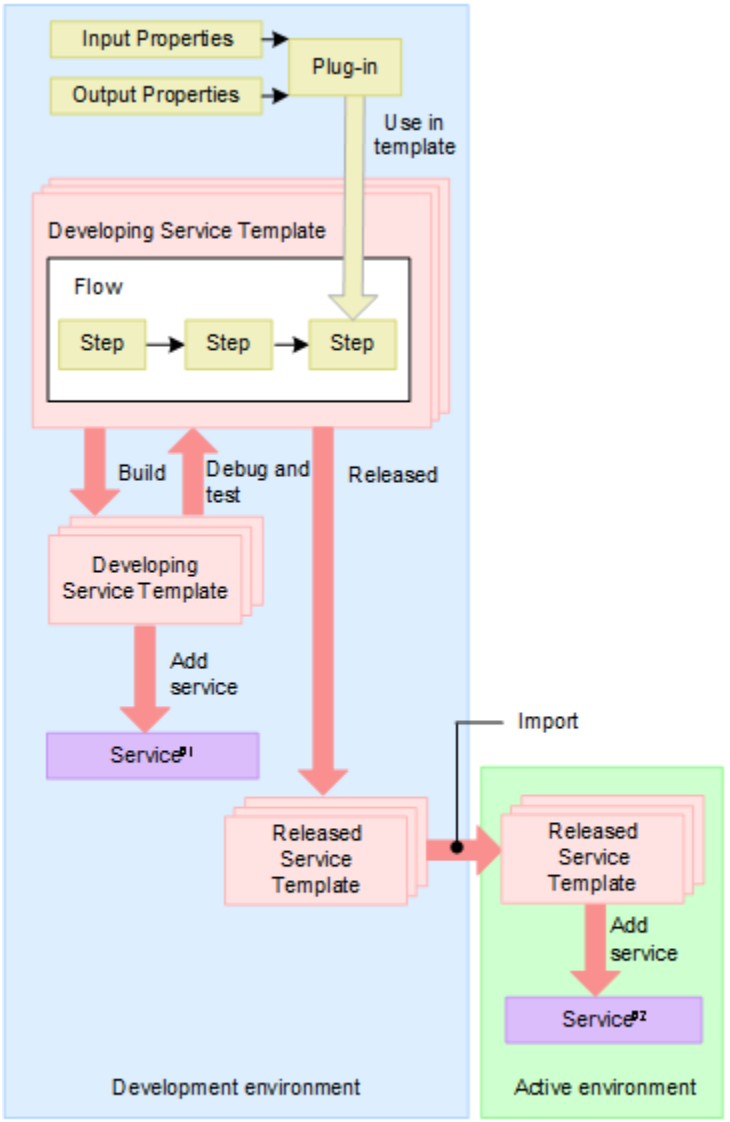
You must have the Ops Center Automator Develop or Admin role to use Service Builder.

For more information on permissions and user roles, see the *Hitachi Ops Center Automator User Guide*.

Terms and concepts

Before beginning to use Service Builder, you must be familiar with the terminology that is used to refer to the various components and have an understanding of how they are interrelated.

The following shows the various Service Builder components and their relationship.



#1
A service for testing the service template in the development environment.
#2
The service running in the active environment.

The following are definitions for each term:

Development and active environments

You create and modify service templates in the development environment until they are ready for testing. After you debug and test a service template, you can import it to the active environment in which the tasks run when a user submits it as a service.

Service template

A service template defines the various operational procedures that you specify through the components (plug-ins and other service templates). Ops Center Automator gives standard service templates that you create, modify and manage with Service Builder. When creating or modifying service templates, you specify the steps and flow and supply the relevant parameters in the development environment until the template is fully debugged and tested. After a service template is functioning properly and completes the tasks without errors, you can import it to the active environment where it is made available as a service.

Developing service template

Developing service templates are used in the development environment. A service template is considered in development when you are specifying the operational parameters and logic through the component steps, during the testing, and until the template is released. Service templates you create by copying a service template are also classified as developing service templates.

Released service template

You use released service templates in the active environment. A service template is released when its associated plug-ins are defined, thoroughly tested, and released for use in the Active environment. The standard service templates provided with Service Builder are also classified as released service templates. After a service template is released, you cannot edit it directly, but you can create a copy of the template and edit it during the development process.

Component

A component is a service template or plug-in that you can add as a step to a flow.

Plug-in

A plug-in is a fundamental component of a service template. It consists of script files, definition files, resource files, and an icon file to complete specific tasks. A service template can contain multiple plug-ins that are linked together to complete a series of tasks.

Input/Output properties

Input properties specify the input settings and values required when running a task and the output properties store the task results that you can use to confirm success or generate errors. You can enter values into plug-in input properties directly, or pass values to them by linking them to a service input property or variable. By linking a service output property to a plug-in output property, you can review the results of a plug-in. Linking properties in this way and passing values between them is called property mapping.

Flow

The flow defines the processing sequence tasks and is established by the placement of the steps and the connectors that link them.

Step

The step is the run unit of a flow that is run based on the placement of the plug-in or service template component. Components are run at each step of the flow. When the step runs, it shows property information.

Service

A service is a custom set of automated instructions and returned output values that are defined by the flow of the component steps and their associated input and output properties in a service template. Services are generated from a standard or customized service template that you create using Service Builder.

Task

A task is the completed instance of a service. You configure and run tasks based on a schedule.

Build/Release

After all the processes are defined and configured through Service Builder, you start the build so that all the required files are packaged into a functioning service template that you can test and debug. After a service template is tested and debugged, you can release it as a service.

Import/Export

After you release a service template, you can export it to a file so that you can import it into another Ops Center Automator installation where it can then be submitted as a service.

Access Service Builder

You create, edit, and manage the service templates that perform a service from the Service Builder **Home** window.

To access Service Builder, go to the Tools menu, then click Service Builder.

Note: You must have an Admin or Develop role to view and access the Tools menu.

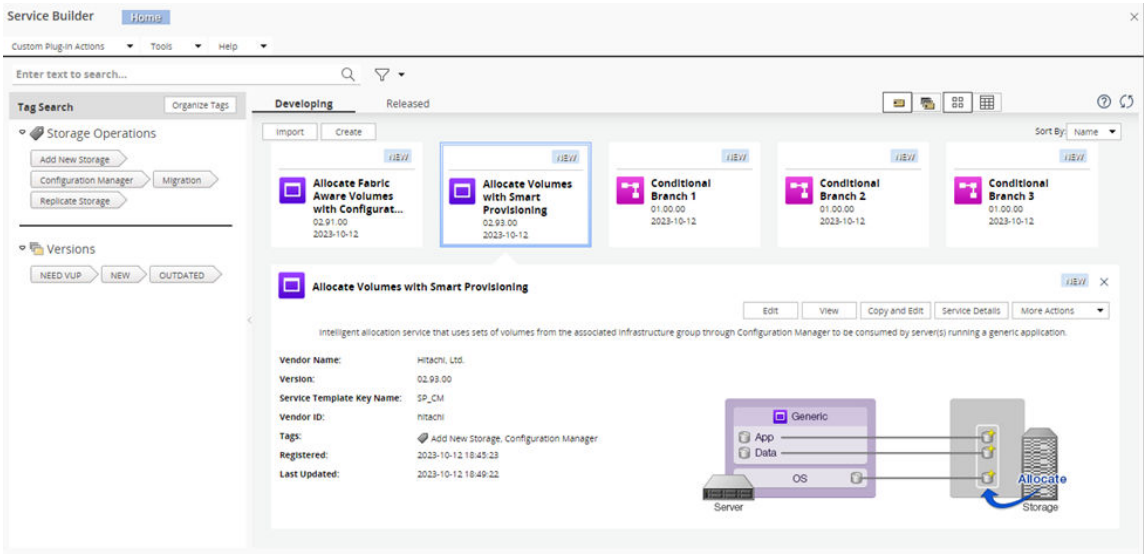
For more information on how to log on to Ops Center Automator and other areas of the UI, see the *Hitachi Ops Center Automator User Guide*.

Navigate the interface

The Service Builder user interface (UI) consists of the following windows and menu options.

Service Builder **Home** window

From the Service Builder **Home** window, you can search for and manage existing service templates and plug-ins and create and edit new ones.



The Service Builder **Home** window shows the service templates that were already created (Released) or those that are currently being worked on (Developing). From here, you can quickly access relevant details on a selected service template, perform management functions (view, copy, edit, delete, import, and export), and create new templates. It is also from the Service Builder **Home** window that you access the options available for working with and creating custom plug-ins that serve as the individual building blocks on which the functions of the service templates are built.

The text search box enables you to search for a service template or you can complete a search on the tag grouping with which the service template is associated. You have the option of showing the service templates using the Card View or Table View.

To sort the service template list, select Sort By, and choose one of the following options:

- Name
- Vendor Name
- Version
- Description
- Service Template Key Name
- Vendor ID
- Registered
- Updated
- Latest Version

Managing and creating service templates

In many cases, you can use one of the existing service templates with some modification to complete the tasks required for your site. If you cannot use any of the standard service templates with modifications, you can create a new one.

The following commands are provided for managing and creating service templates:

- Import: Imports one of the existing service templates that is not automatically installed or one that was created and saved on another system.
- Create: Creates a new service template which can include the properties, commands, and scripts for your site.
- Edit: Enables editing on the selected service template.
- View: Shows details for the selected service template.
- Copy and Edit: Copies the selected service template and then makes the new copy available for editing.
- Service Details: Shows all relevant details for the selected service template.
- More Actions: The following actions are available:
  - Export: Exports the contents of the selected service template to a file.
  - Delete: Deletes the selected service template.

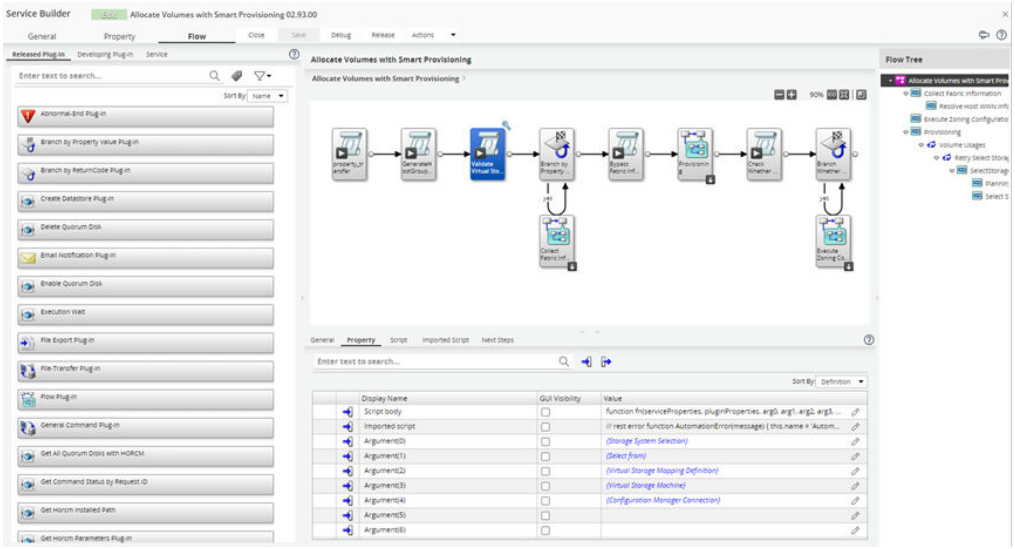
Managing and creating plug-ins

You manage and create plug-ins by choosing the options from the Custom Plug-in Actions menu:

- Create: Creates a new plug-in.
- Edit: Modifies an existing plug-in that is in the Developing state.
- Copy: Copies an existing plug-in that is in the Released or Developing state.
- Delete: Deletes a plug-in that is in the Released or Developing state.

Service Builder **Edit** window

From the Service Builder **Edit** window you can edit templates, where you can select the required plug-ins (or other service templates) and place them in a flow where the logic and values required to complete a task are supplied through input and output properties.



By clicking the associated tab, you can view and edit details and property settings for a service template:

- The General Tab shows general details on the selected service template. By clicking Edit, you can edit and customize the service template.
- The Property Tab shows the input and output properties associated with the service template.

By clicking the Add menu, you have the following options for the service template:

- Property Group: Adds new property groups to categorize various types of properties.
- Input Property: Creates and edits new input properties for the service template.
- Output Property: Creates and edits output properties for the service template.
- Variable: Creates a new variable.
- Service Share Property: Gives access to a collection of commonly used service share properties.

From the Preview menu, you can generate a preview that simulates how a property is processed for a mode of execution (from the **Create Service** window, **Create Request** window, or **Task Details** window) when a service is run.

The More Actions menu has the following additional options:

- Set Visibility: Controls whether the settings are visible when a user runs a service template.
- Set Display Settings: Specifies the following display settings:
  - Editable: Allows the display settings to be edited.
  - Read only: Sets the display settings to read-only.
  - Display: Shows the display settings.
  - Hide: Hides the display settings.
- Flow tab: Gives the following windows:
  - The window on the left shows the available components (Released Plug-in, Developing Plug-in, and Service) that you can to flow of the service template.
  - The upper right window shows the plug-ins and service templates associated with the currently selected service template.
  - The lower right window shows the input and output properties available for the selected component step and general details for the step.
    - General: Gives general details on the currently selected component (plug-in step) and enables you to edit this information. Any Next Step Condition associated with the selected step is also shown.
    - Property: Shows the input and output properties associated with the selected component, which you can edit.
    - Next Steps: Allows the creation of conditional branches that affect how the next step in the flow runs.
  - The Flow Tree window shows a structured view of the component steps that are currently added to the service template.

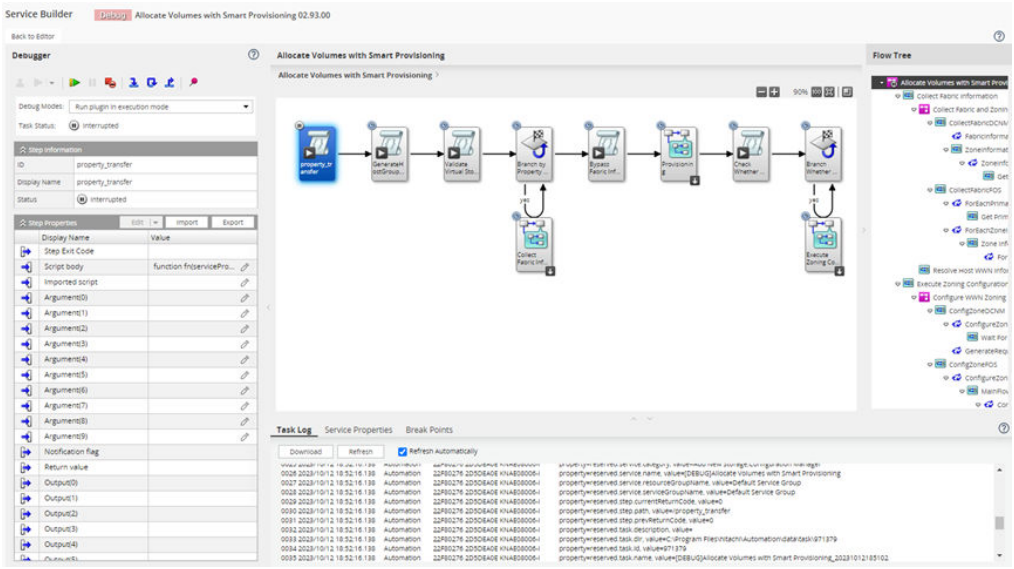
Note: By clicking the airplane icon (in the upper right of the Service Builder **Edit** window, when the Flow tab is selected), you can start a tour of the general process of creating a service template. You can also access an overview of steps in the flow by clicking the Mini Map icon in the upper right of the flow window.

Additional commands at the top of the window provide the following:

- Close: Returns to the Home window.
- Save: Saves the service template in its current state.
- Debug: Builds the service template and gives access to the debugger interface used to run through the tasks and simulate the results to make sure everything is functioning properly.
- Release : Releases the debugged version of the service template where it can then be submitted as a service by a user through the main Ops Center Automator user interface.
- Actions: Gives access to Component Version Management where you can manage component versions.

Service Builder **Debug** window

From the Service Builder **Debug** window you can verify the flow and debug a service template.



The upper center pane shows the flow in which the plug-ins (or other service templates) run and the lower pane shows details depending on which of the following tabs you click:

- Task Log: Shows a log of completed tasks that can be downloaded (Download) and updated on command (Refresh) or automatically (Refresh Automatically) on a regular basis.
- Service Properties: Shows the input and output properties defined for the service template.
- Break Points: Shows all of the breakpoints set for the current debugging session.

The pane on the far right shows a hierarchical view of the task steps and the pane on the far left shows the debug controls.

About the idle timeout

If you log in with Ops Center credentials, and use the UI after the configured idle timeout period has elapsed, you see a session expired error message and have to log in again.

Therefore, when the idle timeout is enabled (Auto-refresh setting in Ops Center Common Services is disabled and the client.editor.sso.timeout.disable property value is false), save the data appropriately before the idle timeout occurs in Service Builder. Note that you can avoid the idle timeout in Service Builder by setting the client.editor.sso.timeout.disable property value to true. See "Changing the system configuration" in the *Hitachi Ops Center Automator Installation and Configuration Guide* for details.

For details about the idle timeout settings, see the *Hitachi Ops Center Online Help*.

Getting started tips

To assist you in initially learning how to accomplish some of the more complicated procedures, Service Builder shows an overlay on the window that visually guides you through the steps.

For example, when you first start to build a service template from the Flow tab, a helpful Getting Started Tips overlay shows how you can search for plug-ins, drag and drop a selected plug-in into the flow, and how to access the properties associated with the plug-in.

If you need it again, you can activate the hints by clicking the airplane icon (in the upper right of the Service Builder **Edit** window, when the Flow tab is selected).

Chapter 2: Working with existing service templates

You manage existing service templates by choosing a template option from the Service Builder **Home** window. This module describes how to view, copy, edit, export, import, and delete existing service templates.

Service template overview

A custom service template consists of plug-ins, (or other service templates) that run the commands or scripts to automate sets of tasks. The plug-ins and services are added as steps and arranged in a sequence to form the operational flow. The service template also needs the mapping of input and output properties to define the flow. Use property groups and service share properties to assist you with defining input and output properties.

Two versions of service templates are available when you access the Service Builder **Home** window:

Note: When creating a new service template, the service output property **`service.errorMessage`** is added to the service template by default.

- Developing: A new service template begins with a debug version. Testing a service template includes creating services and tasks based on the debug version of the service template. The service template resides in the Developing status and can be copied and modified. If a debug version of a service template is built again, the previous debug version of the service template and its services are deleted, and the related tasks are archived. When a Developing version of a service template is released, the debug version of the service template and all the related services are deleted, and the tasks are archived.
- Released: A released version of a service template has completed testing and is available under the Released status. New services and tasks can be created and run based on a released service template. Released service templates cannot be edited, but can be copied and modified for use in another service template.

You can click the Developing or Released tabs from the Service Builder **Home** window to access the service templates. By clicking and highlighting a service template, either from the Card View or Table View, you can access details and complete management functions (view, edit, copy, delete, import, or export) for a service template. If necessary, you have the option of creating a new service template, either based on one of the existing templates or from scratch, and then customizing it for your data center.

Manage existing service templates

You can manage existing service templates, such as viewing, editing, copying, deleting, and so on, from the Service Builder **Home** window.

The procedures for managing existing service templates are covered in the topics in this section.

Viewing a service template

You can view service templates that are in the Developing or Released state from the Service Builder **Home** window.

To view an existing service template, follow these steps:

Procedure

1. From the Service Builder **Home** window, select and highlight the service template from the **Table View** or **Card View** to view the available actions.

   Details on the selected service template and the available options are shown.

2. Click **View**.

Result

The Service Builder **View** window appears with the Flow tab selected, showing the flow for the selected service template. You can click the General and Property tabs to view more details on the selected service template and its associated properties.

Copying a service template

You can copy and edit a service template to create a new version of an existing service template that you can then modify for your data center.

To copy an existing service template, follow these steps:

Procedure

1. From the Service Builder **Home** window, select and highlight the service template from the **Table View** or **Card View** to view the available actions.

   Details on the selected service template and available options are shown.

2. Click **Copy and Edit**.
   The **Copy Service Template** dialog box appears.
3. Enter the basic information for the service template, then click **OK**.
   A copy of the service template is created.

Result

The Service Builder **Edit** view appears with the Flow tab selected, showing the flow for the new service template. You can click the General and Property tabs to view more details on the selected service template and its associated properties.

Copy Service Template dialog box

The dialog box shows the details for a newly copied service template.

The following table describes the **Copy Service Template** dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

Field	Subfield	Description
Key Name: *	-	Specifies the key name for the copied service template.
Version *	-	Version of the service template.
Vendor ID: *	-	Vendor ID.
Display Name: *	-	Name of the service template that is visible on the user interface.
Vendor Name:	-	Vendor name, if applicable, for the copied service template.
Description:	-	Description of the copied service template.
Tags:	-	Tag category associated with the service template.
An asterisk (*) indicates a required field.		

Editing a service template

You can edit an existing service template that was copied and is in the Developing state to modify or add details.

Before you begin

- Identify which components are required to accomplish the purpose of the service template.
- Locate the required components (Released or Developing plug-ins or other service templates) so that they are available for use in the service template.

Procedure

1. From the Service Builder **Home** window, select and highlight the template to edit from the **Developing** tab.
2. Click **Edit**.
   The Service Builder **Edit** window appears with the **Flow** tab selected showing the components (Released or Developing plug-ins and other service templates) associated with the template. You can click the **General** or **Property** tabs to edit additional details on the selected service template.

Next steps

Continue to edit the service template by adding or modifying components, providing the input and output step properties, and establishing the flow.

Deleting a service template

You can delete any service template that is in the Developing state.

To delete a service template in the Released status, run the `deleteservicetemplate` command.

For instructions on running commands in Ops Center Automator, see the *Hitachi Ops Center Automator Installation and Configuration Guide*.

Before you begin

Before you delete a service template in the Developing state, complete the following steps:

1. Stop all the running tasks related to the service template.
2. Archive all the tasks related to the service template.
3. Delete all the services related to the service template.

Note: You must stop and archive all tasks related to the service template and delete all services related to the service template before deleting the service template. You cannot recover a deleted service template.

For instructions on performing these steps, see the *Hitachi Ops Center Automator User Guide*.

Procedure

1. From the Service Builder **Home** window, select and highlight the service template from the **Table View** or **Card View** to view the available actions.

   Details on the selected service template and the available options are shown.

2. From the **More Actions** menu, choose **Delete**.

   The **Delete** confirmation dialog box appears.

3. Click **OK** to confirm deletion.

   The Information menu appears indicating whether the template was deleted.

4. Click **OK**.

Result

The service template is deleted.

Importing a service template

You can import a service template to make use of its service on another system.

Before you begin

To import a service template, the template must be accessible from the local system or the network. You can export a service template from another system to make it accessible for import.

Note: In addition to the service templates that are provided by default, a collection of other service templates are available for import. These templates are in the released state and can be used immediately without having to go through the build process.

To import a service template, follow these steps:

Procedure

1. From the Service Builder **Home** window, select the **Developing** tab, then click **Import**.

   The **Import Service Template Package** dialog box appears.
2. Click **Browse** and specify the name and location of the template to import.
3. Click **OK**.

Result

The service template is imported.

Note: When importing a service template package, a service component in the service template is imported as a service template.

Import Service Template Package dialog box

You can export a service template from one system and then import it to another system.

The following table describes the dialog box fields, subfields, and field groups for the **Import Service Template Package** dialog box. A *field group* is a collection of fields that are related to a specific action or configuration.

Field	Subfield	Description
File Name	-	Click Browse to specify the location and file name of the service template to import.

Exporting a service template

You can export a service template to make its service available on another system.

Before you begin

The service template to export must be accessible on the local system or the network.

Procedure

1. From the Service Builder **Home** window, select and highlight the service template from the **Table View** or **Card View** to view the available actions.

   Details on the selected service template and the available options are shown.

2. Click **More Actions** and from the menu, choose **Export**.
   The **Export** menu appears.
3. Enter the name and location for the exported service template, then click **OK**.

Result

The service template is exported.

Chapter 3: Working with existing plug-ins

Plug-ins are the fundamental building blocks for creating a service template. Each plug-in is designed for a specific purpose and their use and sequence in a service template determines the order in which specific tasks are run. You can use existing plug-ins or edit them to meet your needs.

Plug-ins overview

One or multiple plug-ins (or other released service templates) can be inserted in a service template to run a command or script. Input and output properties and remote commands are set in a plug-in. When using the remote command of a plug-in, the input property can be passed to a command or a script by specifying an input property as the argument of a command or script. Plug-ins can be arranged in a service template to create the flow. Plug-ins are tested and released with the service template to which they are assigned. In addition to the plug-ins, you also have the option of adding other service templates to the flow of a service template.

Types of plug-ins

The two types of plug-ins are as follows:

- Released: Released plug-ins include the custom plug-ins that were released in a service template. When a development service template is released, plug-ins included in the service template become released plug-ins. Released plug-ins cannot be edited, but can be copied and modified for use in other service templates. Released plug-ins are listed under the Released Plug-in tab that is available from the Flow tab of the Service Builder **Edit** window.
- Developing: New plug-ins and plug-ins that have not completed the build process and testing are in the Developing state. Plug-ins in the Developing state can be copied and modified to use in other service templates. During the creation or testing phase, plug-ins in the Developing state are found under the Developing Plug-in tab that is available from the Flow tab of the Service Builder **Edit** window.

In addition to the plug-ins, you can add service templates as components in the flow of another service template:

- Service: A service component is a released service template that has been imported to Ops Center Automator. When a service is used as a component and placed in the flow of another service template, the new service template can incorporate the flow of the service component. Ops Center Automator has a set of built-in service components with its built-in services. Service components are found under the Services tab that is available from the Flow tab of the Service Builder **Edit** window.

Custom Plug-in List dialog box

The **Custom Plug-in List** dialog box shows the plug-ins you can access for an option, (such as copy, edit, delete, and so on) from the Custom Plug-in Actions menu. You can search for an existing plug-in by entering some identifying text or by the tags associated with the plug-ins. You can manage existing plug-ins from the Released tab or access plug-ins that you are currently working on from the Developing tab. You also have the option of specifying how the plug-ins are listed and visible in the window by clicking either the Card View or Table View.

The following table describes the dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

Field	Subfield	Description
Vendor Name	-	Vendor name, if applicable, for the plug-in.
Version	-	Version of the plug-in.
Key Name	-	Key associated with the plug-in.
Vendor ID	-	Vendor ID associated with the plug-in.
Tags	-	Tag category associated with the plug-in.
Registered	-	Date and time when the plug-in was registered.
Last Updated	-	Date and time when the plug-in was last updated.

Manage existing plug-ins

You can manage existing plug-ins from the Service Builder **Custom Plug-in List** dialog box by choosing an option from the Custom Plug-in Actions menu.

Copying a plug-in

You can make a copy of a plug-in and modify it for your data center.

You can copy any plug-in that is in the Developing or Released state. When copying an existing plug-in, you must assign a new vendor ID, plug-in key name, or version number.

Procedure

1. From the Service Builder **Home** window, access the **Custom Plug-in Actions** menu and choose **Copy**.
   The dialog box appears.
2. Select and highlight the plug-in to copy either from the **Released** or **Developing** tab, then click **Copy and Edit**.
   The dialog box appears.
3. Enter the required plug-in information from the **General**, **Property**, or **Remote Command** tabs, then click **Save**.
   A copy of the plug-in is created and is available from the **Developing** tab of the **Custom Plug-in List** dialog box.

Next steps

If necessary, continue working with the copied plug-in by accessing the Custom Plug-in Actions menu and selecting Edit.

Editing a plug-in

You can edit the input and output properties, variables, and remote commands associated with a plug-in that is in the Developing state.

Procedure

1. From the Service Builder **Home** window, select **Custom Plug-in Actions** > **Edit** .
   The **Custom Plug-in List** dialog box appears in which you can select the plug-in to edit.
2. Select the custom plug-in to edit, then click **Edit**.
   The **Edit Custom Plug-in** dialog box appears for the plug-in.
3. Edit the selected plug-in by choosing the settings from the associated tabs (**General**, **Property**, or **Remote Command**). Click **Save** when you complete the editing.
   An informational message appears indicating that the edited version of the plug-in was saved.

Copy Custom Plug-in dialog box

Use the **Copy Custom Plug-in** dialog box to provide the details when copying a plug-in.

The following tables describe the dialog box fields, subfields, and field groups that are available based on the tab that is currently selected. A *field group* is a collection of fields that are related to a specific action or configuration.

From the General tab, you can view and enter the following details on the selected plug-in.

Field	Subfield	Description
Key Name: *	-	Specifies the key name for the copied plug-in.
Version *	-	Version of the plug-in.
Vendor ID: *	-	Vendor ID assigned to the plug-in.

Field	Subfield	Description
Display Name: *	-	Plug-in name that is shown in the user interface.
Vendor Name:	-	Vendor name, if applicable, for copied plug-in.
Description:	-	Description of the copied plug-in.
Tags:	-	Tag category associated with the plug-in.
Icon:	-	Icon associated with the plug-in.
Fields marked with an asterisk (*) are required.		

When making a copy of a plug-in, change the Key Name or Vendor ID to differentiate the copied version from the original.

From the Property tab, you can search for and view all the input and output properties associated with the selected plug-in. You also have the option of managing the properties by clicking one of the options. For example, you can add a new input or output property by choosing Add, edit an existing property by selecting Edit, or delete a selected property by selecting Delete.

Field	Subfield	Description
Key Name	-	Specifies the key name for the input or output property.
Display Name	-	Specifies the display name for the input or output property.
Description	-	Shows a description of the input or output property and its function.
Required	-	Indicates if the input or output property is required (true) or not (false).
Default Value	-	Indicates any default value that is associated with the input or output property.

From the Remote Command tab, choose the platform from which to run the remote command by selecting an option from the Add Platform menu. You can also specify the Credential Type by choosing either Shared agentless setting or Service input property. Other options for controlling how to run the remote command depend on the selected platform.

Field	Subfield	Description
Credential Type	-	Specifies the credential type required for the plug-in.

Field	Subfield	Description
		Select Shared agentless setting if you use the credential information in the agentless remote connections view under the Ops Center Automator Administration tab when the service is run. Shared agentless setting is the default Credential Type. The following reserved plug-in property is automatically set for the shared agentless setting credential type: plugin.destinationHost Enter the target of a process by IPv4 address, IPv6 address, or host name (up to 256 characters). Select Service input property to use the credential information as an input property. The following reserved plug-in properties are automatically set for the Service input property credential type: ▪ plugin.destinationHost Enter the target of a process by IPv4 address, IPv6 address, or host name (up to 256 characters). If the destination host is Ops Center Automator Server (localhost), the user ID and the password are not necessary. ▪ plugin.account Enter the user ID for logging on to the target host (up to 256 characters). ▪ plugin.password Enter the password for logging on to the target host (up to 256 characters). ▪ plugin.suPassword Enter the password of the root account used for logging on to a target host in a Linux OS environment (up to 245 characters). If the target host is running on the Windows OS, this property is ignored.
Windows Options	Run as system account	Runs the command using the system account.
Linux/UNIX Options	Execute with root privileges	Runs with root privileges.

Field	Subfield	Description
	Character Set Auto Judgment	Character Set Auto Judgment applies to the Linux OS. If enabled, the script runs using the default locale of the user. If disabled, the script runs with the LC_ALL=C locale. The default is enabled.
Add Platform	-	Specifies the platform for the remote command (Windows or Linux platform).
Import Settings	-	Specifies that the settings are imported from another OS.
Execution Method:	-	Select Script name if based on a script, or CLI Command if using a stored command.
CLI Command: *	-	Enter the CLI command (up to 8,192 characters) for the remote command. Required if Script name or CLI Command is selected for Execution Method. Insert All Input Properties inserts all input properties defined for the plug-in to the command.
Script specification method:	-	Select Attachment if uploading a script file, or Type in to directly enter the script information. Use the following file format for a plug-in that runs a script: *`name-of-plug-in.extension`*.
File*	-	Click Browse to specify the file that holds the command or script.
Mapping Definition of Output Properties	-	Specify the mapping definition of the output property for a command or script. By selecting and highlighting a property in the list and then clicking the pencil icon, you can access the **Edit Output Filter** dialog box where you can specify an output filter that controls the data that is passed to the output property.
Execution Directory	-	Specify the folder from which the command runs.
Environment Variables	-	Select environment variables for the command or script. When selecting this option, the **Create/Edit Environment Variable** dialog box appears where you can enter the required variable information.
Fields marked with an asterisk (*) are required.		

Deleting a plug-in

You can delete a plug-in you no longer need. You cannot recover a deleted plug-in.

Procedure

1. From the Service Builder **Home** window, select the **Released** or **Developing** tabs. Access the **Custom Plug-in Actions** menu and choose **Delete**.
   The **Custom Plug-in List** dialog box appears.
2. Click and highlight the plug-in to delete from either the **Card View** or **Table View** and then click **Delete**.
   The **Delete** confirmation dialog box appears.
3. Click **OK** to confirm deleting the plug-in.

Result

The plug-in is deleted and is no longer shown in the Service Builder **Home** window.

Chapter 4: Creating a new service template

Creating a new service template involves creating a new template or copying and editing an existing template. The new or modified service template is then defined by the components (plug-ins or other service templates), resource files, icon files, custom files, flows, and service definitions.

Service template creation workflow

You can manage and create new service templates from the Service Builder **Home** window. (You also can manage and create new plug-ins from the Custom Plug-in Actions menu.)

To manage an existing service template (edit, view, copy, import, or export), select the service template (either from the Developing or Released tab), then click the action you must complete. For the details on a service template, click Service Details.

When creating a new service template, you must do some planning and then complete the phases described in the following workflow:

Phase 1: Preparation

1. Decide the purpose of the service template. Consider the steps for automating the process and determine if you must create a new template or modify an existing template.
2. Prepare to create the service template. This involves identifying existing components, or creating new components, preparing icon files, and setting definition files, resource files, custom files, and flow.

Phase 2: Creation

1. Create a new service template or copy and modify an existing service template. Enter the basic information such as the name, ID, vendor name, and description. The service template is now in the Developing state.
2. Create new plug-ins or use the existing plug-ins if they are sufficient.
3. Define the flow for the service template by arranging the templates and plug-ins as steps and then connect them in the order you need them to run.
4. Define the data flow by mapping the input and output properties of the component with the input and output properties of the service template.
5. Set the service definitions with the input and output properties of the service template. Add service share properties, property group information, and variables.

Phase 3: Testing/Debugging

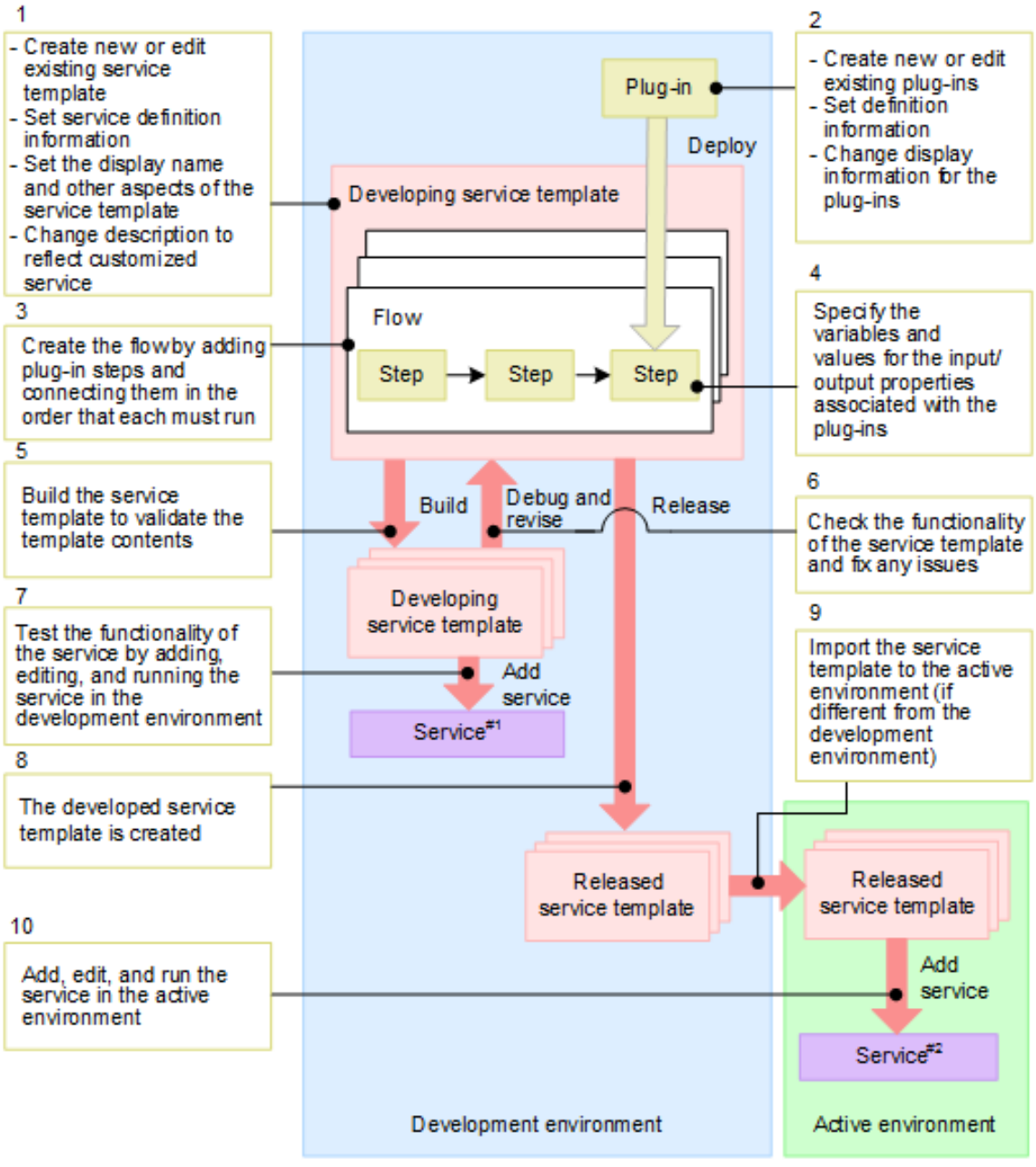
1. Build the service template for testing.

2. Complete testing. Create services based on the service template debug configuration.
3. Make corrections as the result of testing.
4. Rebuild and retest the service template until it runs successfully.

Phase 4: Releasing

Release the service template. A service template must be in the Released status to submit it to the active environment.

The following figure shows the typical steps for creating or editing a service template.



#1
A service that tests the template in the development environment
#2
The service running in the active environment

Creating a new service template

You can create a new service template to create a custom service that automates a series of tasks for a user.

Before you begin

- Identify the components required to accomplish the purpose of the service template.
- Locate the required components (Released or Developing) so that they are available for use in the service template.

Note: Although the procedure for creating a new service template is described, in many cases you can copy an existing service templates to use as a model and then make the required modifications.

Procedure

1. Choose one of the following methods to create a new service template:
   - To create a new service template from scratch, from the Service Builder **Home** window, access the **Developing** tab, then click **Create**.

     The **Create Service Template** dialog box appears.
   - To create a new service template based on an existing template, from the Service Builder **Home** window, access either the **Developing** or **Released** tab, then click **Copy and Edit**.

     The **Copy Service Template** dialog box appears.
2. Enter the information for the new service template (and select any tags you want the template associated with), then click **OK**.
   The Service Builder **Edit** window appears with the **Flow** tab selected.

Result

You can begin searching for and dragging and dropping the required components (Released or Developing plug-ins and other Service Templates) to the flow in the order you need them to run.

Next steps

Continue to provide the input and output step properties, establish the flow and, when done, debug and release the service template.

Create Service Template dialog box

When creating a new service template, supply the template details from the **Create Service Template** dialog box.

The following table describes the dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

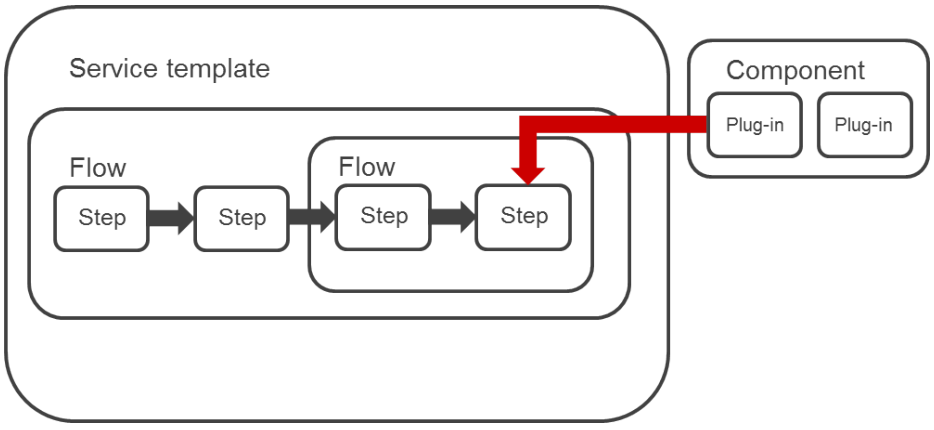
Field	Subfield	Description
Key Name *	-	A unique key name is assigned to each service template for easy access and tracking.
Version *	-	A version number for the new service template.
Vendor ID *	-	An ID associated with the vendor.
Display Name: *	-	Name of the new service template that is shown through the user interface.
Vendor Name:	-	Vendor name, if applicable.
Description:	-	Description of the new service template.
Tags:	-	Specifies any tags categories the template is associated with.
Fields with an asterisk (*) are required.		

Specify the step flow

You can specify the steps for a service template by adding the components (plug-ins or other service templates) and establishing the run order from the Flow tab.

When establishing the flow of steps in a service template, you must add the plug-in (or service template) steps from the Flow view and then draw connectors to establish the run order. You then specify the mapping for the input and output properties.

The following figure shows the typical flow for a service template.



When you first start to build a service template from the Flow tab, a helpful "Tour" option leads you through the plug-in search process, how to drag and drop a selected plug-in to a flow, and how to access the properties associated with the plug-in. After initially viewing this tour, you can specify that it not be shown again by checking the associated check box. You can always activate the tour again.

Creating the steps in a data flow

You create the steps in a service template from the Service Builder Flow tab of the **Edit** window by specifying the required components for a service.

Before you begin

- A service template must be open and in the Developing status for editing or creation.
- Identify the plug-ins required to accomplish the purpose of the service template.
- Locate the components (Released or Developing) so that they are available for use in the service template.

Procedure

1. From the **Service Builder Edit Flow** tab, click a component, drag it to the **Flow** view, and release.
   The **Create Step** dialog box appears.
2. Enter the information for the step, then click **OK**.
   The component icon appears in the **Flow** view as a step and step details, including related properties, are accessible from the **General** or **Property** tabs.
3. Continue to add steps as required.
   - To edit, copy, cut, or delete a step, right-click the component and select the associated option. When copying a component, you can paste it in another service template.
   - To select multiple steps, drag the mouse to create a rectangle encompassing the targeted steps, or click each step icon while pressing the **Ctrl** key to add to the selection.
   - To move a step, click the target step icon, drag it to the targeted area of the **Flow** view, and release.
   - To add a second flow, right-click the target flow in the **Flow Tree** view, then click **Create Flow**.
4. To exit the Service Builder **Edit** window without saving, click **Close**. To save your changes, click **Save**.

Result

Steps are added and saved to the service template.

Next steps

Continue to edit the service template and add a flow that shows the run order of the steps.

Create/Edit Step dialog box

You can enter the details associated with a component for a step in a service template from the **Create/Edit Step** dialog box.

Note: You can click View to get details on a step component.

The following table describes the dialog box fields, subfields, and field groups for the **Create/Edit Step** dialog box. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

Field	Subfield	Description
Step	Step ID*	The ID of the step.
	Step Name*	The name of the step.
	Description	A short description of the step.
Component	Vendor Name	The vendor name associated with the component.
	Version	The version of the plug-in. Click the version list to view the choices and see if there are multiple versions.
	Key Name	The key name associated with the component.
	Vendor ID	The vendor name of the plug-in.
	Tags	The tag categories associated with the step.
	Registered	The date and time the step was originally registered.
	Last Updated	The date and time the step was last updated.
Next Step Conditions	Condition:	Sets instructions for when to run the next step. The choices are: ▪ Determine the return value based on the threshold Run the next step when the return value is equal to or less than the Error Threshold. If the return value is more than the Error Threshold, this step ends with an error status. ▪ Always succeed regardless of return value Always run the next step. ▪ Always fail regardless of return value Always end after this step with an error status.
	Error Threshold:*	Set a whole number from 0 through 255 that establishes the threshold value at which a conditional step runs.
	Use Warnings:	If enabled, runs the next step with an error status when the Use Warnings are exceeded.
	Warning Threshold:	Required if the Flow condition in case of error is enabled. Set a whole number from 0 through 255. If the return value is less than the Warning Threshold and the Error Threshold, run the next

Field	Subfield	Description
		step. If the return value is equal to or more than the Warning Threshold, but equal to or less than the Error Threshold, the task status reflects "In Progress (with Error)" when running and "Failed" when the task is complete. The Warning Threshold number cannot be larger than the Error Threshold.
An asterisk (*) indicates a required field.		

Specifying step properties

You must specify the input and output properties that are used for the tasks associated with a component (plug-in or other service template).

A service template defines a generic operating procedure. For this reason, properties that store the input values required to run the service, such as host names and resource limits, are defined when services are added from a service template. These are called service input properties. The results of running a service are output to the Ops Center Automator user interface as the values of service output properties. Input properties that store the input values required for running a step and output properties that store results are defined through the plug-ins. You can enter values in plug-in input properties directly, or pass values to them by linking them to a service input property or variable. By linking a service output property to a plug-in output property, you can review the run results of a plug-in from the Ops Center Automator user interface. Linking properties in this way and passing values between them is called *property mapping*.

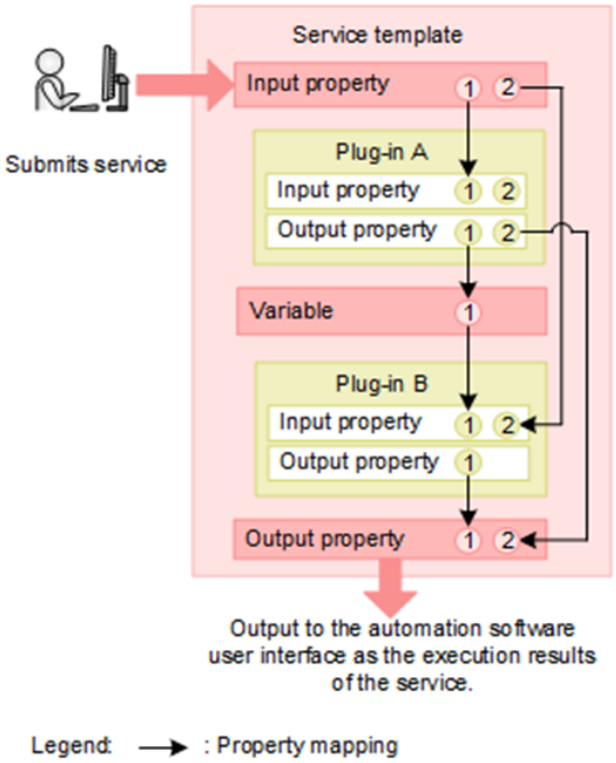
You must map the component input and output properties for every step with the input and output properties and variables of the service template. Associate the input properties of the components used in a service template with the input properties or variables, or output properties of other component steps associated with a service template. The service template input properties store the input values required to run a service. The service template output properties store the run results of the service.

The value of the plug-in input property can be the value that was previously set for the property, or directly set for the service template. The input property can also be a variable. You can apply the service share properties to the input properties.

The output property content depends on the type of component. You can store the results of running a step as output properties. Variables temporarily hold the values that are passed between components.

The input and output properties of the components are set from the Property tab in the Service Builder Edit window. The input and output properties and variables for the service template are set in the Property tab of the Service Builder **Edit** window.

The following figure shows a typical mapping between the service template properties and the corresponding plug-in step properties.



In this figure:

- The value input to Input property 1 of the service is input to Input property 1 of Plug-in A.
- The value input to Input property 2 of the service is input to Input property 2 of Plug-in B.
- The unmapped Input property 2 of Plug-in A is assigned the value entered when the service template was created or edited.

Because Output property 1 of Plug-in A is mapped to Variable 1, and Variable 1 is mapped to Input property 1 of Plug-in B. Values output as Output property 1 of Plug-in A are stored in Variable 1 then input to Input property 1 of Plug-in B. This passes the run results of Plug-in A to an input property of Plug-in B, so it can be used in the processing of Plug-in B. Because of this mapping, the following results are achieved:

- The run results of Plug-in A (standard command output and standard error output, and output properties) output as Output property 2 of Plug-in A are also output to Output property 2 of the service. This enables you to review the run results of Plug-in A in the Ops Center Automator user interface.
- The run results of Plug-in B (standard command output and standard error output, and output properties) output as Output property 1 of Plug-in B are also output to Output property 1 of the service. You can also review the results of Plug-in B through the Ops Center Automator user interface.

Before you begin

A service template in the Developing state must exist with steps you added to the Flow view.

Procedure

1. From the Service Builder **Flow** tab of the **Edit** window, click and highlight a step in the flow.
   A list of properties associated with the selected component (plug-in or other service template) are shown from the **Property** tab. You can either view the input properties or the output properties by clicking the icon next to the search box that allows you to enter a text search to locate specific properties in a long list.
2. With the input properties icon selected, enter the input properties for the step:
   - Enter the details directly in the **Value** column of the table.
   - Click the **pencil** icon to enter the details using the dialog box.
3. With the output properties icon selected, click the **pencil** icon to enter the details of the output properties using the **Specify Component Output Property Mapping Parameters** dialog box and then click **OK**. You can add new output properties by clicking **Add Output Property** and you can add variables by clicking **Add Variable**.
4. Continue to enter the required input and output properties.
5. If necessary, you can verify the **GUI Visibility** check box that determines whether the selected property is visible to Modify/Submit users.

Next steps

Set the service definitions of the service template.

Specify Component Input/Output Property Mapping Parameters dialog box

You can specify the mapping of input and output properties for the components incorporated in a service template through the **Specify Component Input/Output Property for Mapping Parameters** dialog box.

The following table describes the dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

Field	Subfield	Description
Key:	-	The input or output property key.
Display Name:	-	Name of the input or output property.
Description:	-	Description of the input or output property.
Data Type:	-	Data type of the property: string, boolean, integer, double, date, password, or composite. Available options depend on the options you chose. You can verify the Array Type option for the data type to be treated as an array. In this way, a set of the properties of the same type (number of elements is a variable) can be handled as a single property to make data mapping easier, especially when passing data between a Service and the Plug-ins.
Setting Method:	-	Setting method can be View Property or Direct Input. When entering a direct input value, you click Insert Property to make the value from the specified property the value.
Value:	-	Direct input value associated with a property.

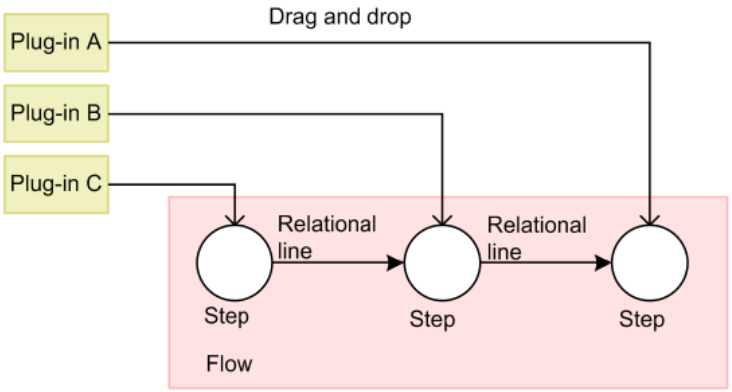
When specifying component input properties, you can view a list of plug-in steps shown under the **Step Tree** for the currently selected component.

Depending on whether you are working with input or output properties, you can click Add Input Property or Add Output Property to add an input or output property, or Add Variable to add a variable to a property group.

Establishing the flow of execution

Map the process flow of a service template from the Service Builder **Edit** window Flow tab. Use the Flow view to connect the steps icons and establish a flow.

You establish each unit of processing in a flow by dragging plug-ins from the Plug-in view to the Flow view. Each plug-in dropped to the Flow view is called a step. Create a flow by placing the steps required to run a task in the required order and connecting them with relational lines as shown in the following figure.



The flow can contain one step that connects to two or more steps and similarly, two or more steps can connect to one step. In the figure, the next step is run only after every connected step finishes. You can also use the Flow plug-in and Repeated-execution plug-in to define a flow in another flow.

Follow these steps to establish the flow of steps for a service template:

Before you begin

A service template in the Developing state must exist with the steps you added to the Flow view.

Procedure

1. From the Service Builder **Edit** window, access the **Flow** tab, and then drag and add the necessary plug-ins (or services) into the **Flow** view in the relative order the steps are run by the service.
2. To establish the order in which the steps in the flow are run, click and hold the **node** associated with the step that to run first, drag the connector line across to the step to run next, and then release.
   A connection line appears indicating the steps are connected. The arrow indicates the direction of the flow.
3. Continue to add connections to steps as needed.
   - To delete a connection, click the targeted connection line, then click **Delete**.

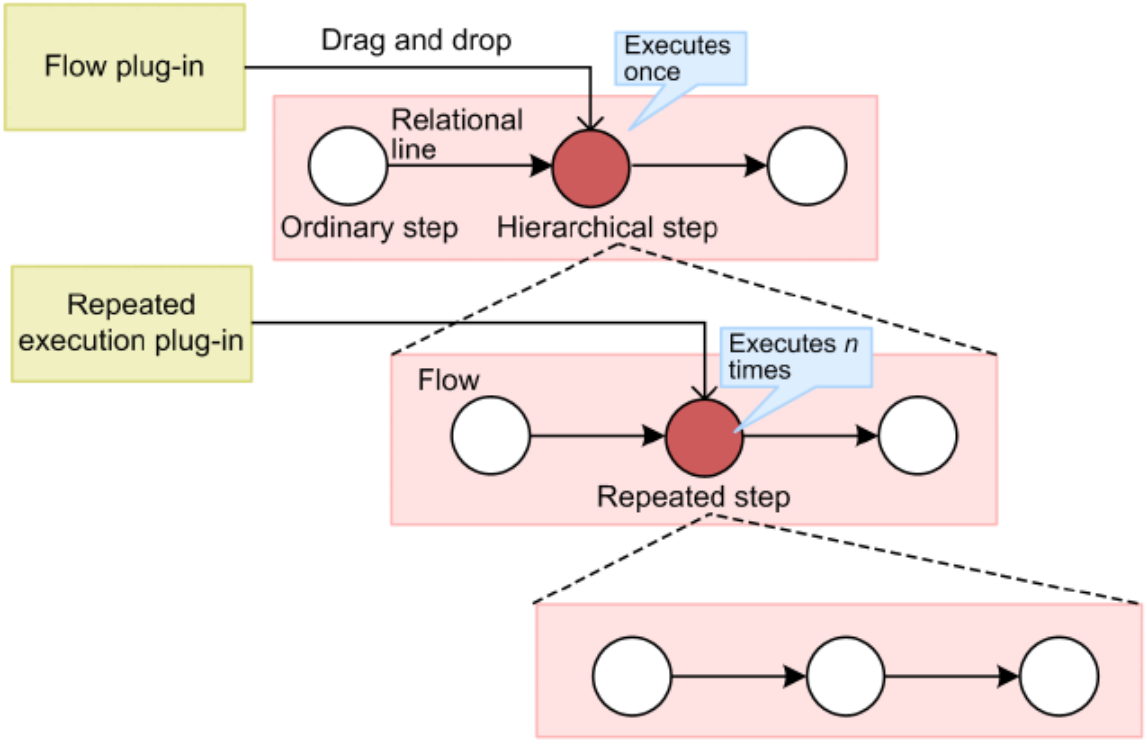
Next steps

Define the input and output properties and enter the mapping parameters.

Creating a flow hierarchy

You can create a flow hierarchy in a service template by defining a flow in another existing flow. The Flow Tree view shows the hierarchy of the flow.

You also can create a flow hierarchy by deploying flow plug-ins, and repeating a unit of processing that consists of several steps by deploying repeated-execution plug-ins as shown in the following figure.



The following table shows the plug-in roles and their relationship to the various steps in the flow.

Dragged and dropped plug-in	Type of step	Role
Flow plug-in	Hierarchical step	Creates a flow hierarchy.
Repeated Execution plug-in	Repeated step	Repeats execution of the specified flow. To create a hierarchy on a flow subordinate to the repeated step, you must use a flow plug-in. If you try to create a hierarchy on a flow you copied and pasted that includes a repeated step, an error occurs. Use the Repeated Execution plug-in to create a nested loop to a maximum of three levels.
Other plug-ins	Ordinary step	Runs the plug-in normally.

You can view a list of the steps in a flow from the Flow Tree view where it shows the name of the service template as the first step of the flow. Lower levels are represented by the step name associated with the step that runs the flow plug-in or repeated-execution plug-in. When you run a service template that includes a hierarchical flow, only the top-level steps in the flow appear in the **Task Details** dialog box. Steps in subordinate flows and in flow plug-ins and repeated-execution plug-ins are not visible.

Before you begin

A service template in the Developing state must exist with steps you added to the Flow view.

Procedure

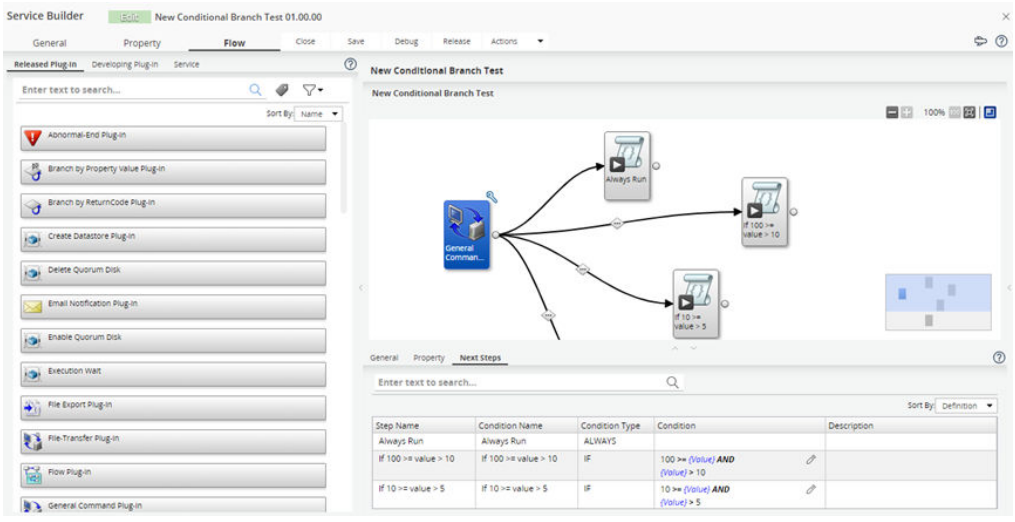
1. From the Service Builder **Edit** window, select the **Flow** tab, and then drag the required plug-ins to the **Flow** view to establish the execution hierarchy for the other plug-ins in the flow.
2. To repeat execution of the specified flow, select the Repeated-execution plug-in and drag it into position.
3. Add connections to steps as needed.
4. (Optional) To delete a connection, click the targeted connection line, then click **Delete**.

Creating a Next Step conditional branch in a flow

You can create a conditional branch to control the flow of steps that run depending on when a condition is met.

A Next Step conditional branch is useful for running a step in a flow based on a condition.

You specify next step conditions from the Next Steps tab when establishing the flow in a service template.



The step names and conditional settings associated with any of the steps in the flow are shown:

Step Name

Name of the step for which the condition is defined.

Condition Name
Name of the conditional expression to evaluate. By default, this is the next step name.

Condition Type
Type of condition (ALWAYS, IF, OTHER) that must be met for a next step to run. When selecting [OTHER], there must be an item with a conditional expression of type [IF].

Condition
Specifies the condition using valid expressions that are applied to service properties.

Description
Gives a brief description of the condition being evaluated.

To add a Next Step condition:

Procedure

1. Drag the step to include in the conditional flow to the flow area of the window.
2. Connect a line between the preceding step in the conditional flow and the next step to run if the condition occurs.
3. From the **Next Steps** tab, click the required Condition Type (ALWAYS, IF, OTHER) from the list. If you choose ALWAYS, the step always runs normally by default. Because you must use IF and OTHER together, your only other choice is IF.
4. Click the pencil icon to access the **Specify Execution Condition** dialog box where you can enter the value for the condition.
5. Enter the condition value, then click **OK**. Optionally, you can also enter a description for the next step condition.

   When you finish, the condition is indicated by a conditional icon on the execution flow line arrow.

   To create a more complex conditional branching, you can use multiple Branch by Returncode Plug-ins so that one step is run when a condition is met and another step is run when the condition is not met.

Specify Execution Condition dialog box

You can specify the steps that run next in a flow based on a condition.

The following table describes the dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

Name	Description	Editability
Step Name	Name of the step for which the condition is defined.	Not Allowed
Condition Name	Name of the conditional expression that is to be evaluated. By default, this is the next step name.	Allowed

Name	Description	Editability
Description	Brief description of the condition that is being evaluated.	Allowed
Value	Specifies the condition using valid expressions that are applied to service properties.	Allowed

Following are the valid operators.

Symbol/String	Meaning	Notes
OR	Logical OR	▪ The operator can be uppercase or lowercase. ▪ A space is required on either side of the OR operator. ▪ A mixing of logical AND is not allowed. But, logical OR is allowed.
AND	Logical AND	▪ The operator can be uppercase or lowercase. ▪ A space is required on either side of the AND operator. ▪ A mixing of logical OR is not allowed. But, logical AND is allowed.
=	Equal sign	▪ The operator can be a string or a numeric value. ▪ A space is required on either side of the = operator. ▪ A full-width number is considered a string.
!=	Not equal to sign	▪ The operator can be a string or a numeric value. ▪ A space is required on either side of the != operator. ▪ A full-width number is considered a string.
<=	Less than or equal	▪ Only numerical values are allowed. ▪ A space is required on either side of the <= operator. ▪ A full-widths number is not allowed.

Symbol/String	Meaning	Notes
>=	Greater than or equal	▪ Only numerical values are allowed. ▪ A space is required on either side of the >= operator. ▪ A full-width number is not allowed.
<	Less than	▪ Only numerical values are allowed. ▪ A space is required on either side of the < operator. ▪ A full-width number is not allowed.
>	Greater than	▪ Only numerical values are allowed. ▪ A space is required on either side of the > operator. ▪ A full-width number is not allowed.
equals	Equal sign	▪ Only string values are allowed. ▪ The operator can be uppercase or lowercase. ▪ A space is required on either side of the "equals" operator.
not equals	Not equal to sign	▪ Only string values are allowed. ▪ The operator can be upper or lower case. ▪ A space is required on either side of the "not equals" operator.
contains	Contain	▪ Only string values are allowed. ▪ The operator can be uppercase or lowercase. ▪ A space is required on either side of the "contains" operator.
not contains	Does not contain	▪ Only string values are allowed. ▪ The operator can be uppercase or lowercase. ▪ A space is required on either side of the "not contains" operator.

Symbol/String	Meaning	Notes
\	Escape	▪ Interprets and distinguishes an operator symbol as a string. ▪ Insert the escape operator before each word in an expression that is to be treated as a string (for example, "a \not \equals b"). ▪ If treating a double quotation mark (") as a string, specify (\"). ▪ If treating a backslash as a string, specify (\\).

Specify the property settings

You specify the input and output properties, service share properties, and variables for a service template from the Service Builder **Edit** window Property tab. These settings affect how the various parameters associated with a template appears to the user.

From the Property tab, you complete the following tasks:

- Open/Close All Groups: Displays or collapses the view of the property groups.
- Add: has the following options:
  - Property Group: Adds a property group for organizing the display of properties.
  - Input Property: Creates a new input property for the selected service template.
  - Output Property: Creates a new output property for the selected service template.
  - Variable: Creates a new variable for the selected service template.
  - Service Share Property: Adds a service share property.
- Edit: Edits details for the selected property.
- Delete: Deletes the selected property.
- Preview : Previews the use of the property from the Create Service Window, Create Request Window, or Task Details Window.
- More Actions: Has the following options:
  - Set Visibility: Specifies whether these items are visible to the user from the Create Service window only or the Create Service and Create Request windows.
  - Set Display Settings -- Sets the display settings for modification by the user as Editable, Read only, Display, or Hide.

In some cases, you must customize the service template by changing the default icon or the text and figures that display for the Service Details dialog box and the overview associated with the service.

Selecting the service share properties

You can add service share properties to a service template to implement common, predefined functions.

Before you begin

A service template in the Developing state must exist with the steps you added to the Flow view.

Procedure

1. From the Service Builder **Edit** window, access the **Property** tab.
2. From the **Add** menu, choose **Service Share Property**.
   The **Select Service Share Property** dialog box appears. A list of service share properties that you can assign to the property group are provided for the current service template.
3. Select the service share property to add to the service template from the list, then click **OK**.
   The selected service share property is added to the service template.

Next steps

- To set the value for the selected service share property, access the **Edit Input Property for Service** dialog box.
- You can add a service share property to a property group by accessing the Add menu, selecting Property Group, and supplying the required attributes.

Select Service Share Property dialog box

You select the service share properties for a property group from the **Select Service Share Property** dialog box.

The following table describes the **Select Service Share Property** dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

Field	Subfield	Description
Display Name:	-	Display name assigned to the selected property.
Key:	-	Key name associated with the service share property.
Description:	-	Description of the function implemented by the service share property.
Property Group:	-	Specifies the property group to which the service share property is assigned.

You can set values for service share properties by clicking Edit.

Select Reference Property dialog box

You select reference properties from the **Select Reference Property** dialog box.

The following table describes the **Select Reference Property** dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

Field	Subfield	Description
Step Tree Area	-	Show the list of steps defined in a template. Information in the Step Tree area is highlighted by the following icons: ▪ RESERVED PROPERTY: ▪ Service Template: ▪ Service component: ▪ Flow plug-in: ▪ Repeated Execution plug-in: ▪ Others:
Properties List	-	Shows the Display Name and Key Name of properties defined for the group selected in the Step Tree area. Initially, only the Step directly before the Step with the property that you are editing is visible.
Show All Steps	-	When set to ON, all the steps defined in the template are visible. The initial display is OFF. Not all steps are shown when the Param Mode of the Property you are editing is off.
Reserved Properties:	-	When the Repeated Execution Plug-in is selected in the Step Tree area, the reserved properties related to the plug-in are visible. This item is visible only when the Repeated Execution Plug-in is selected in the Step Tree area.

The selectable repeated execution input value (reserved.loop.input, reserved.loop.input*N*) and repeated execution loop index (reserved.loop.index, reserved.loop.index*N*) are shown in a menu in one line. To help you easily understand the relationship between repeated execution input value and repeated execution loop index and the corresponding Repeated Execution Plug-in, the step name of the Repeated Execution Plug-in corresponding to the Display Name is also described in parenthesis.

Following are the reserved properties:

Property	Display Condition	Display Name and Format
reserved.loop.input	When selecting a Repeated Execution Plug-in that is one level above the base step as seen from the plug-in where this dialog box appears in the Step Tree area.	Input for repeated execution 1 level above (*corresponding step name*)
reserved.loop.index		Loop index 1 level above (*corresponding step name*)
reserved.loop.input*N*	When selecting a Repeated Execution Plug-in that is *N* levels above the base step as seen from the plug-in where this dialog box appears in the Step Tree area.	Input for repeated execution *N* levels above (*corresponding step name*)
reserved.loop.index*N*		Loop index N levels above (*corresponding step name*)

Adding input properties

You can specify the input properties for a property group associated with a service template.

Before you begin

A service template in the Developing state must exist with steps you added to the Flow view.

Procedure

1. Add input properties to a property group associated with a service template by using one of the following methods:
   - From Service Builder **Edit** window, access the **Property** tab, go to the **Add** menu, and choose **Input Property**.
   - From the **Flow** tab **Step Properties** area, go to the **Value** field, then click the pencil icon. The **Specify Component Input Property Mapping Parameters** dialog box appears in which you can add properties. Click **Add Input Property**.
2. The **Create Input Property for Service** dialog box appears.
3. Enter the details for the input property.
4. Click **OK** to save the input property details.
   The specified input property appears in the input property list.

Create/Edit Input Property for Service dialog box

The following table describes the **Create/Edit Input Property for Service** dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

Table 1 Definitions Field Group

Field	Subfield	Description
Key: *	-	Input property key name.
Display Name: *	-	Name of the input property.
Description:	-	Description of the input property.
Property Group:	-	Select the property group to which the property belongs. You can also choose Create New Property Group to create a new property group.
Visibility:	-	Choose whether properties are visible on both the **Edit** and **Submit** windows, or only on the **Edit** window.
Display Settings:	-	Specifies the display setting for the input property. The choices are: ▪ Editable ▪ Read only ▪ Hide
Service Share Property:	-	Enable Service Share Property to add the service as a Service component after the release process.
Required:	-	Specifies that the property is required when this check box is checked.
Data Type:	-	Select a data type of the property: string, boolean, integer, double, date, password, composite. Various options that specify restrictions on the data entry are visible depending on the option you chose. When using arrays, verify the Array Type option for the data type to be handled as an array. By doing this, a set of the properties of the same type (number of elements is variable) can be handled as a single property making data mapping easier, especially when passing data between a service and the plug-ins.
Content Type:	-	Select the content type: ▪ application/json ▪ application/javascript ▪ application/xml

Field	Subfield	Description
		▪ text/html ▪ text/plain ▪ text/csv ▪ application/octet-stream
Domain Type:	-	Select the domain type from the list or add a new domain type by clicking Add New Domain Type and entering the details in the **Create Domain Type Definition** dialog box. This option is available when you choose composite for the Data Type and application/json for the Content Type.
Fields with an asterisk (*) are required.		

Table 2 Restrictions Field Group

Field	Subfield	Description
Minimum Value:	-	Specifies the minimum value. This input field is visible when you choose integer, double, or date for the Data Type.
Maximum Value:	-	Specifies the maximum value. This input field is visible when you choose integer, double, or date for the Data Type.
Minimum Length:	-	Specifies the minimum length of the property. This input field is visible when you choose string or password for the Data Type.
Maximum Length:	-	Specifies the maximum length of the property. This input field is visible when you choose string or password for the Data Type.
Restricted Character:	-	Specifies the allowed characters by using a regular expression. This input field is visible when you choose string or password for the Data Type. Example: ^[0-9a-zA-Z\.\-]*$
Minimum Array Length:	-	Specifies the minimum length of array elements.
Maximum Array Length:	-	Specifies the maximum length of array elements.

Field	Subfield	Description
Validation Script:	-	Script that validates the property based on the associated JavaScript code.

Table 3 Value and Presentation Field Group

Field	Subfield	Description
Presentation:	-	Specifies the property presentation. The available presentation values appear in the list depending on the Data Type.
Default Value:	-	Specifies whether the default value for the property is true or false. When specifying the Array of option, the default value must be written as a comma-separated string value surrounded by brackets. Example: ["1","2","3"]
Data Source:	-	Specifies whether the data is Static or Dynamic and obtained from an external resource provider.
Specify List Items:	-	Specifies the Specify List Items when the data source for the property is static (when you choose the Static option for the Data Source.)
External Resource:	-	Specifies the external resource provider when the data source for the property is dynamic (when you choose the Dynamic option for the Data Source). In the list, you can also add, edit, upload, or delete for the external resource provider.
Extra Path:	-	Specifies the extra path portion of the request URL. Leave it empty if it is not required. The extra path is the path that follows the external resource provider ID in a URL as follows: /Automation/v1/objects/ExternalResources/ *<external resource provider ID>*/*<extra path>*? *<query parameters>*

Field	Subfield	Description
Query Param:	-	Specifies the query parameter for the external resource provider. The `serviceID` and `serviceTemplateID` parameters are added automatically. You can specify {$ref:*keyName*} to refer to the value of other properties in the same property group. For a JSON value, you can specify {$ref:*keyName#json path*}.
Name Field:	-	Specifies the field name to use as the label visible in the list. If omitted, uses the name field.
Value Field:	-	Specifies the field value to use as the value of the property visible in the list. If omitted, uses the instance ID field.
Show If:	-	Specifies to show the property entry if the specified conditions are met.
Enable If:	-	Enables the property if the specified conditions are met.

Creating a validator script for verifying an input property

If the validation options are not adequate, you can create a script. Following is an example of a validator script written in JavaScript code that verifies whether a value entered by the user is a number less than the maximum allowable value of 1024:

```
function(propertyValue, lang, displayType){
  var jObject = JSON.parse(propertyValue.value);
  if(displayType == "config"){
    if( jObject.luSize > 10){
      return "lu size should be under 10"
    }
    if( jObject.blockSize > 2){
      return "block size should be under 2";
    }
  }

 return
}
```

The following table shows the validator script specifications for the input property.

Name	Description
Validator script format	function (arg1, arg2, arg3) {

Name	Description
	//code }
Validator script arguments	arg1: Property value in string format arg2: Locale string. (for example, ja or en) arg3: Operating information when script is running (Operation with task creation: exec, Editing operation of properties: config)
Validator scrips return value*	Success: undefined, null Failure: Error message in array or string format
* If the value is not a number or is larger than the specified maximum, a message appears in the user interface.	

Using external resource data for input properties

When you specify "Selection" or "Radio Button" as the input property presentation, you can use the external resource data listed in the menu. You can create the external resource provider by using Service Builder to set up and access the external resource data.

Create an external resource provider by using Service Builder

You can create a new external resource provider by clicking Add New External Resource Provider at the bottom of the list. You can also specify an existing external resource provider from the list.

In the **Create External Resource Provider** dialog box, enter the following information:

- Name: Specifies the name of the external resource provider.
- Version: Specifies the version number.
- Content Type: Selects either application/json or text/csv. For the input property where "Radio Button" is specified as the presentation, text/csv is not supported.
- Schema ID: Specifies the schema ID of the domain type corresponding to the external resource provider. This helps users select an external resource provider in the property for service and plug-in setting dialog boxes; in the list of external resource providers, those with the same schema ID as the selected domain type are highlighted in blue.

- Description: Specifies a description for the external resource provider.
- Type: Select either Javascript, Script, Command Line, or File. Depending on your selection, enter the required information in the field.

When specifying Javascript for the type

The following table shows the JavaScript specifications for the external resource provider.

Name	Description
Script format	function fn (requestPath, queryParamMap, properties) { // code }
Arguments of script	requestPath: A value of Extra Path which is specified in the Create/Edit Input Property for Service dialog box. queryParamMap: A JSON object including key-value maps which is specified as Query Param in the Create/Edit Input Property for Service dialog box. properties: A JSON object including service share properties and reserved properties which is related to external resource provider. The reserved properties are the following: ▪ reserved.external.hcmds.dir ▪ reserved.external.path ▪ reserved.external.query ▪ reserved.external.resource.dir ▪ reserved.external.userName You can obtain the value of the property using properties["*property key*"].
Return value of script	Return an array of JSON objects which are listed in the menu. The array to be returned must be set to a property named "data".

In the script, you can use the `auto util library` as the utility function.

Using the auto util library

You can use the `auto util library`. See the `auto util library` in JavaScript Plug-in (on page 230) for more information.

The following shows a code sample when running the Configuration Manager REST API in an external resource provider.

```
function fn(requestPath, queryParamMap, properties) {
  /** This is sample code that calls the Configuration Manager REST API. */
```

```
  //Step 1. Generate a method to run REST API to Configuration Manager.
  var configurationManagerCall = function(request) {
    var respBody = null;
    auto.util.http.handleCall(auto.util.storage.restCall, request,
      function(resp, req) {
        respBody = auto.util.parseJson(resp.responseBody);
      }, function(resp, req) {
        auto.util.http.defaultErrorHandler(null, req, resp);
      }, function(err, req) {
        auto.util.http.defaultErrorHandler(err, req);
      }, auto.util.storage.retrySettings
    );
    return respBody;
  };

  //Step 2. Get accessible Web Service Connections by specifying a category name.
  var wsc = auto.util.env.getWebServiceConnections("ConfigurationManager");

  //Step 3. Get Session
  //You can pass arguments by using the plug-in input properties or you can specify
them directly in a script. For example, you can pass the device ID through the query
parameters (queryParamMap).
  var request = {
    "requestMethod": "POST",
    "requestUrl": "/ConfigurationManager/v1/objects/storages/" +
queryParamMap.deviceId + "/sessions",
    "requestHeaders": auto.util.http.toRawHeader({}),
    "authScheme": "basic",
    "connectionName": wsc[0].name,
    "productName": wsc[0].productName
  };
  var respBody = configurationManagerCall(request);
  var token = respBody.token;
  var sId = respBody.sessionId;

  //Step 4. Call the API that is associated with your use case based on the session
obtained in Step 3.
//When specifying the Authorization header for the request in the user program,
specify "none" for authScheme.
  var headers = {
    "Authorization": "Session " + token,
  };
  var queryMap = {
    "poolType" : "DP"
  };
  var queryStr = auto.util.http.buildQuery(queryMap);
  request = {
    "requestMethod": "GET",
    "requestUrl": "/ConfigurationManager/v1/objects/storages/" +
queryParamMap.deviceId + "/pools" + queryStr,
```

```
    "requestHeaders": auto.util.http.toRawHeader(headers),
    "authScheme": "none",
    "connectionName": wsc[0].name,
    "productName": wsc[0].productName
  };
  var pools = configurationManagerCall(request);

  //Step 5. Make an array containing the required information.
  //An array to be returned has to be set to a property named "data".
  var ret = { "data": [] };
  var p;
  for (var i = 0; i < pools.data.length; i++) {
    p = pools.data[i];
    ret.data.push({
      "Pool ID": p.poolId,
      "Pool Type": p.poolType,
      "Num of LDEVs": p.numOfLdevs
    });
  }

  //Final Step. Discard Session.
  request = {
    "requestMethod": "DELETE",
    "requestUrl": "/ConfigurationManager/v1/objects/storages/" +
queryParamMap.deviceId + "/sessions/" + sId,
    "requestHeaders": auto.util.http.toRawHeader(headers),
    "authScheme": "none",
    "connectionName": wsc[0].name,
    "productName": wsc[0].productName
  };
  configurationManagerCall(request);

  return ret;

}
```

When specifying Script for the type

Python is supported as a script type. Specify the path to the Python interpreter that runs the script, and edit a script. The supported versions of Python are version 3.x series. To use this external resource provider in a cluster environment, the Python interpreter must be installed on both the active and standby systems. This external resource provider does not support a virtual Python environment.

Environment variables that can be referenced from the script

The following table shows the environment variables that are allowable in the script. You can get the values for the following environment variables in the `os.environ[`*`key-name`*`]` or `os.environ.get(`*`key-name`*`)` format.

Environment variable	Description	Format
REQUEST_PATH	Information specified as Extra Path in the Create/Edit Input Property for Service dialog box	String */external-resource-provider-ID/value-specified-as-extra-path*
QUERY_PARAM_MAP	Information specified as Query Param in the Create/Edit Input Property for Service dialog box	JSON format *{property-name*:*value*, ...}
SERVICE_TEMPLATE_ID	ID of the service template to which the Python plug-in belongs	Numerical value
SERVICE_ID	ID of the service running the Python plug-in	Numerical value
SERVICE_TEMPLATE	Information about the service template to which the Python plug-in belongs	JSON format {*service-template-attribute*:*value*, ...}
SERVICE	Information about the service running the Python Plug-in	JSON format {*service-attribute*:*value*, ...}
STORAGE_PROFILES	Information about the Storage Profile	JSON format [{ *StorageProfile-attribute*:*value, ...* }, ...]
WEB_SERVICE_CONNECTIONS	Settings information for the Web Service Connection. This corresponds to the query parameters ("__webServiceConnectionCategory__" and "__webServiceConnectionName__") specified in Query Param seen in the following table.	JSON format [{ *WebServiceConnection-attribute*:*value*, ... }, ...]

Query Param Parameters		Reference information
__webServiceConnectionCategory__	**__webServiceConnectionName__**	
Parameter is specified. (Y)	Parameter is specified. (Y)	Web Service Connection information that coincides with the specified Category and Name
Parameter is specified. (Y)	Parameter is not specified. (N)	Web Service Connection information that coincides with the specified Category
Parameter is not specified. (N)	Parameter is specified. (Y)	None
Parameter is not specified. (N)	Parameter is not specified. (N)	None

When specifying command line for the type

Enter the command line to run.

- An error is generated if the return value of the command is not 0.
- The upper limit for standard output is 30 MB and an error is generated if this limit is exceeded.
- When specifying a script file, use the absolute path because the current path is not specified.
- The service share property and the reserved property can be included in a command line. If including a service share property or a reserved property, surround the property key with "${" and "}".
- The character set for a standard output assumes the character set of the system and the associated user. (For example, in the case of Windows OS with Japanese Locale, MS-932, and in the case of Linux OS, the user who last ran the start command.)
- In the command line, you can refer to the values for the environment variables. The referable environment variables are the same as when specifying Script for the type. Note that the method of obtaining the value for the environment variable depends on the script language, as shown in the following table.

Item	How to get the environment variable value
Command script (Windows)	%key-name%
Shell script (Linux)	$key-name
PowerShell script	$env:key-name

The following example shows how host names are acquired through a program such as PowerShell, and are then visible in a list as follows:

```
Command line:
powershell.exe \"& '${reserved.external.resource.dir}\\getHosts.ps1' $
{reserved.external.hcmds.dir} ${reserved.external.userName}\"

Output of a command line:
name,instanceID
host1,123
host2,124
host3,125
host4,126
host5,127

The list item displayed on the Config/Submit window of the service:
host1
host2
host3
host4
host5
```

When specifying file for the type

Enter the path of the files. The service share property and the reserved property can be included in a file path. If including a service share property or a reserved property, surround the property key with "${" and "}".

In the following example, the host data is acquired from JSON file and output by using another application.

```
File :
${reserved.external.resource.dir}\\vm.json

vm.json :
{
"data" :[ {
"instanceID" :127,
"name" :"test1"
}, {
"instanceID" :128,
"name" :"test2"
} ]
}

The list item displayed on the Config/Submit window of the service:
test1
test2
```

Upload a file to an external resource provider

You can upload files to an external resource provider by clicking Upload in the external resource provider list. You can update a .zip archive file. After updating files, you can specify the relative path name in the command line as a file path.

Delete an external resource provider

You can delete an external resource provider by clicking Delete in the external resource provider list. When you click Delete, the **Delete Confirmation** dialog box appears. The dialog box lists the service templates and plug-ins using the external resource provider you plan to delete. When you confirm that there is no related service templates and plug-ins, click OK and the external resource is deleted. If you delete an external resource provider that is used by a service template, the external resource provider no longer works with the service template.

Create/Edit Domain Type Definition dialog box

You can edit the schema for a selected domain type by selecting the Domain Type option available from the **Create/Edit Input/Output Property for Service** dialog box and then clicking the plus sign or wrench icon next to the domain type.

The following table describes the **Edit Domain Type Definition** dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

Field	Subfield	Description
Domain Type Name: *	-	Specifies the name for the domain type.
Domain Type ID:	-	Specifies an ID (address) for the domain type.
Domain Type Schema:	-	Specifies the schema for the domain type. The format conforms to the JSON schema (http://json-schema.org/).
Generate Template (Button):	-	Generates a GUI Definition based on the domain type schema. You do not need to enter the GUI Definition from scratch because the basic structure is the same as for the Domain Type ID.
GUI Definition ID:	-	Specifies UI definition ID.
GUI Definition:	-	Specifies the UI definition.
Fields with an asterisk (*) are required.		

The following table shows the property definition attributes in the GUI Definition.

Key	Description
displayName	Display name of property.
description	Description of property.
presentation	Presentation type of property. "input", "textarea", "url", "select", "radio", "check box", "spinbox", "capacity", "capacityInKB", "capacityInMB", "capacityInGB", "capacityInTB", "capacityiB", "capacityInKiB", "capacityInMiB", "capacityInGiB", "capacityInTiB", "datePicker", "hex", or "file".
permission	Permission of property; "locked", "unlocked", or "hidden"
visibility	Visibility of property; "exec" or "config"
required	Specifies whether the property is required; true or false.
pattern	Accepted regular expression pattern of the property value.
validationScript	Validation script of the property value.
showIf	Show condition of property.
enableIf	Activate condition of property.
enum	Specifies an array. You can define JSON objects in it.
enumDataSource	Specifies an external resource provider as in the following example: `"enumDataSource": { "url":"/Automation/v1/objects/ExternalResources/1eb39858-48b5-4d3a-b82a-9af5c91af8c4", "contentType": "application/json", "nameField": "hostGroupName", "valueField": "id" }`
contentType	Content type of composite type; "application/json", "application/javascript", "application/xml", "text/html", "text/plain", "text/csv", or "application/octet-stream"

Key	Description
hidden	Show or hide a column in a table. true: Hide a column by default. false: Show a column by default.

Following are examples of the domain type schema and UI definition representing a volume setting in the built-in Allocate Volumes with Smart Provisioning service:

Domain type schema

```
{
  "type": "object",
  "properties": {
    "volumeUsage": {
      "type": "string",
      "minLength": 1,
      "maxLength": 64,
      "pattern": "^[A-Za-z0-9 ~!@#\\$%\\^&()_\\+\\-=\\{\\}\\[\\]'\\.`]*$"
    },
    "numberOfVolumes": {
      "type": "integer",
      "minimum": 1,
      "maximum": 200,
      "default": 1
    },
    "volumeCapacityInMiB": {
      "type": "integer",
      "minimum": 47,
      "maximum": 268435456
    },
    "blockCapacity": {
      "type": "string",
      "pattern": "^[0-9]+$",
      "validationScript": "function validate(value, language, displayType) {var
errorMessages = [];if (!value || !value.indexOf) {return errorMessages;}if
(Number(value) > 549755813888) {errorMessages.push('The maximum value is
549755813888.');}else if (Number(value) < 96000) {errorMessages.push('The minimum
value is 96000');}return errorMessages;}"
    },
    "volumeLabel": {
      "type": "string",
      "maxLength": 32,
      "pattern": "^[A-Za-z0-9\\.:@_][A-Za-z0-9\\-\\.:@_]*$",
      "validationScript": "function validate(value, language, displayType) {var
errorMessages = []; var re = /^[A-Za-z0-9\\.:@_][A-Za-z0-9\\-\\.:@_]*$/; if (!value
|| !value.indexOf) {return errorMessages;} if (!re.test(value))
{errorMessages.push('You can use only following caracters and white space: a-z,A-Z,0-
```

```
9,-,.,:,@,_')} else if (value.indexOf(' ') === 0) {errorMessages.push('LDEV label
must not start with a white space.');} else if (value.indexOf(' ') === value.length -
1) {errorMessages.push('LDEV label must not end with a white space.');} return
errorMessages;}"
    },
    "diskType": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "ldevSetting": {
      "type": "object",
      "properties": {
        "ldevIdStartsFrom": {
          "type": "integer",
          "minimum": 0,
          "maximum": 16777215
        },
        "virtualLdevIdStartsFrom": {
          "type": "integer",
          "minimum": 0,
          "maximum": 65279
        }
      }
    },
    "lunSetting": {
      "type": "object",
      "properties": {
        "lunStartsFrom": {
          "type": "integer",
          "minimum": 0,
          "maximum": 4095
        }
      }
    }
  }
}
```

GUI definition

```
{
  "properties": {
    "volumeUsage": {
      "displayName": "Volume Usage",
      "required": true
    },
    "numberOfVolumes": {
      "displayName": "Number of Volumes",
      "description": "Specify the number of volumes.",
```

```
      "required": true
    },
    "volumeCapacityInMiB": {
      "displayName": "Volume Capacity",
      "description": "Specify the volume capacity in bytes.",
      "presentation": "capacityInMiB",
      "showIf": "function showIf(properties, language, displayType) { return
( properties.CapacityFormat === \"Byte\"); }",
      "enableIf": "function enableIf(properties, language, displayType) { return
( properties.CapacityFormat === \"Byte\"); }",
      "required": true
    },
    "blockCapacity": {
      "displayName": "Volume Capacity",
      "description": "Specify the volume capacity in blocks. (1 block = 512 bytes)",
      "showIf": "function showIf(properties, language, displayType) { return
( properties.CapacityFormat === \"Block\"); }",
      "enableIf": "function enableIf(properties, language, displayType) { return
( properties.CapacityFormat === \"Block\"); }",
      "required": true
    },
    "volumeLabel": {
      "displayName": "Volume Label",
      "description": "Specify the volume label.",
      "required": false
    },
    "diskType": {
      "displayName": "Disk Type",
      "description": "Specify the disk type of the pool to use.",
      "required": false,
      "items": {
        "presentation": "select",
        "enumDataSource": {
          "url": "/Automation/v1/objects/ExternalResources/8890eb8d-c1fd-4067-954d-
c91f47f4de42",
          "contentType": "application/json",
          "nameField": "name",
          "valueField": "value"
        }
      },
      "showIf": "function showIf(properties, language, displayType) { return
((properties.StorageSelection === \"Automatic\") || (properties.PoolSelection ===
\"Automatic\"));}"
    },
    "ldevSetting": {
      "displayName": "LDEV Setting",
      "properties": {
        "ldevIdStartsFrom": {
          "displayName": "LDEV ID Starts From",
          "description": "Specify the startup LDEV ID as a hexadecimal number for the
volume to allocate.",
```

```
          "presentation": "hex",
          "required": false
        },
        "virtualLdevIdStartsFrom": {
          "displayName": "Virtual LDEV ID Starts From",
          "description": "Specify the startup Virtual LDEV ID  for the volume to
allocate.",
          "presentation": "hex",
          "required": false,
          "showIf": "function showIf(properties, language, displayType) { return
properties.SelectFrom !== 'Virtual Storage Machine' || properties.StorageSelection !
== 'Automatic'; }"
        }
      }
    },
    "lunSetting": {
      "displayName": "LUN Setting",
      "properties": {
        "lunStartsFrom": {
          "displayName": "LUN Starts From",
          "description": "Specify the starting logical unit number assigned to the
volume for a host.",
          "required": false,
          "presentation": "hex"
        }
      }
    }
  }
}
```

Adding output properties

You can specify the output properties for a property group associated with a service template.

A list of properties is shown.

Before you begin

A service template in the Developing state must exist with steps you added to the Flow View.

Procedure

1. From the Service Builder **Edit** window **Property** tab, go to the **Add** menu and choose **Output Property**.
   The **Create Output Property for Service** dialog box appears.
2. You can also add output properties from the **Specify Component Output Property Mapping Parameters** dialog box under the **Flow** tab, by clicking the pencil icon in the **Value** field of the **Step Properties** area of the window.
3. Click **Add Output Property**.

   The **Create Output Property for Service** dialog box appears.

4. Enter the details for the output property.
5. Click **OK** to save the output property details.
   The specified output property appears in the output property list.

Create/Edit Output Property for Service dialog box

You can add output properties for a service template from the **Create/Edit Output Property for Service** dialog box.

The following table describes the Create/Edit Output Property for Service dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

Table 4 Definitions Field Group

Field	Subfield	Description
Key: *	-	Output property key name.
Display Name: *	-	Name of the output property.
Description:	-	Description of the output property.
Property Group:	-	Select the property group to which the property belongs. You can also choose (Create New Property Group) to create a new property group.
Display/Hide:	-	Specifies the display setting for the output property. The choices are: ▪ Display ▪ Hide
Data Type:	-	Select a data type of the property: string, boolean, integer, double, date, password, composite. Various options are specifying restrictions on the data entry are visible depending on the option you choose. When dealing with arrays, you can verify the Array Type option for the data type to be handled as an array. In this way, a set of the properties of the same type (number of elements is variable) can be handled as a single property making data mapping easier, especially when passing data between a service and the plug-ins.

Field	Subfield	Description
Content Type:	-	Select the content type: ▪ application/json ▪ application/javascript ▪ application/xml ▪ text/html ▪ text/plain ▪ text/csv ▪ application/octet-stream
Domain Type:	-	Select the domain type from the list or add a new domain type by clicking Add New Domain Type and entering the relevant details from the Create Domain Type Definition dialog box. This option is available when you choose composite for the Data Type and application/json for the Content Type.
Fields with an asterisk (*) are required.		

Table 5 Value and Presentation Field Group

Field	Subfield	Description
Presentation:	-	Specifies the property presentation. The presentations types are visible in the list depending on the Data Type.
Default Value:	-	Specifies the default value for the property. The values that can be specified differ depending on the Data Type.
Show If:	-	Shows the property entry if the specified conditions are met.
Enable If:	-	Enables the property if the specified conditions are met.

Adding variables

You can add variables for a property group associated with a service template.

A list of properties for a Property Group are visible on the Property tab.

Before you begin

A service template in the Developing state must exist with steps you added to the Flow view.

Procedure

1. From the Service Builder **Edit** window **Property** tab, go to the **Add** menu and choose **Variable**.
   The **Create Variable** dialog box appears.
2. Enter the details for the variable you are adding. You can add a service share property to a property group from the **Property Group** menu. You can also specify the data type and associated default value.
3. Click **OK**.
   The variable is added to the service template.

Next steps

Map the variable to the output value.

Create/Edit Variable dialog box

You can enter or edit a variable property for a service template from the **Create/Edit Variable** dialog box.

The following table describes the dialog box fields, subfields, and field groups for the **Create/Edit Variable** dialog box. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

Table 6 Definitions Field Group

Field	Subfield	Description
Key Name:*	-	Variable key name.
Display Name:*	-	Variable name.
Description:	-	Variable description.
Property Group:	-	"Default Properties" are set. For variables, you cannot change this item.

Field	Subfield	Description
Data Type:	-	Select a data type of the property: string, boolean, integer, double, date, password, composite. The various options depend on which option you choose. For example, if you choose the date option, a calendar interface appears. When using arrays, verify the Array Type option for the data type to be handled as an array. In this way, a set of the properties of the same type (number of elements is variable) are handled as a single property making data mapping easier, especially when passing data between a service and the plug-ins.
Content Type:	-	Select the content type: ▪ application/json ▪ application/javascript ▪ application/xml ▪ text/html ▪ text/plain ▪ text/csv ▪ application/octet-stream
Domain Type:	-	Select the domain type from the list or add a new domain type by clicking Add New Domain Type and entering the details from the **Create Domain Type Definition** dialog box. This option is available when you choose composite for the Data Type and application/json for the Content Type.
Fields with an asterisk (*) are required.		

Table 7 Value and Presentation Field Group

Field	Subfield	Description
Default Value:	-	Enter the default value of the variable.

Example of creating a new service template

This section describes the process of customizing a service template that provisions volumes for a specific platform and adds a component step that generates an email notification indicating whether the volume allocation was successful.

For this example, you complete the following procedures:

1. Make a copy of an existing service template and provide the details for the new service template.
2. Add an email notification plug-in and modify it for your environment.
3. Establish the flow for the component steps.
4. Test the new service template.

Note: The example assumes that you have considered the system's architecture and completed the calculations necessary to create a service based on the required storage size, configuration, and I/O profile. Although the template values are based on the best practices, the values you set depend on your requirements.

To begin this example, you must first access the Service Builder **Home** window by going to the Tools menu and choosing Service Builder. The Service Builder **Home** window shows all the Service Builder options for creating and managing service templates and the associated plug-ins.

Making a copy of an existing service template

You must be a service administrator with the Develop role to complete the following steps:

Procedure

1. From the Service Builder **Home** window **Released** tab, select and highlight the template to copy either from the **Card View** or **Table View**. For this example, select the **Allocate Volumes with Smart Provisioning** service template.
2. Click **Copy and Edit**.

   The **Copy Service Template** dialog box appears.
3. Enter the basic information as shown in the following table.

Parameter	Description	Value
Key Name: *	-	Specifies the key name for the copied service template. For this example, enter a unique ID, such as "NewTemplate".
Version *	-	Version of the service template. This is already filled in.
Vendor ID: *	-	Enter a name that identifies the vendor.

Parameter	Description	Value
Display Name: *	-	Name assigned to the copied version of the service template. This is already filled in. For this example, enter a name such as "NewTemplate".
Vendor Name:	-	Enter a name that identifies the vendor.
Description:	-	Description of the copied service template.
Tags:	-	Tag category associated with the service template. The "Add New Storage" and "Configuration Manager" tags are already selected.
Fields with an asterisk (*) are required.		

4. Click **OK** and the new service template is created with the modified details. The components (plug-ins and possibly other service templates) associated with the copied template are visible on the Service Builder **Edit** window **Flow** tab.
   The existing plug-in is already added as a step to the flow from the original service template that you copied. If you click this step, you can verify the input and output properties associated with the step.

Adding email notification for the service template

After you make a copy of the service template, you can make the necessary modifications from the Service Builder Edit window.

To add the Email Notification Plug-in, complete the following steps:

Procedure

1. From the component list, locate the **Email Notification Plug-in** and drag it to the flow area of the window on the right. If necessary, you can use the search box to locate the plug-in. The **Create Step** dialog box appears.
2. Enter the required details for the plug-in step, as shown in the following table.

Parameter	Description	Value
Step ID: *	-	Specifies the ID for the **Email Notification Plug-in**. This is already filled in.
Step Name: *	-	Name assigned to the plug-in step. This is already filled in.
Description:	-	Optional description of the copied service template.
Fields with an asterisk (*) are required.		

3. Click **OK** to add the selected plug-in. A graphic for the newly added plug-in step is shown in the flow area of the window.
4. Specify the run order for the newly added plug-in step by clicking the dot next to the existing plug-in graphic and dragging the arrow over the **Email Notification Plug-in** step. An arrow shows the direction in which the service template processes the plug-in steps when you run the service.
5. Specify the input properties for the newly added plug-in step. For example, you can enter a subject for the Subject property or add content for the message body. Because the values for email notification are different for each user, you might want to verify the **GUI Visibility** check box for the "To Addresses" property so a user can enter a relevant email address from the **Edit** or **Submit** windows when running the service.

   To enter a relevant email address from both the **Edit** and **Submit** windows, change the visibility for the "To Addresses" property to "Edit and Submit Window" from the **Property** tab.

Debugging, building, testing, and releasing the new service template

After creating and adding the required plug-ins, you can build the service template, test and debug it, and then release it.

Procedure

1. Click the **Debug** tab and then click **OK** to build the service template with the added plug-ins.

   The **Build / Release Result** dialog box appears showing any errors that occurred while building the service template.
2. Continue to troubleshoot any errors until the build completes successfully, then click **Close** to exit the **Build / Release Result** dialog box.
   If the build is successful, the **Perform Debugging** dialog box appears.
3. Verify that the service template is functioning correctly. To accomplish this, you must return to the main Ops Center Automator UI where you can create a request and submit the service request for the service template.
4. On the **Services** tab:
   a. Click **Create**.
   b. Click **Show All Versions** on the **Select Service Template** window to show "NewTemplate".
   c. Select "NewTemplate", then click **Create Service**.
   d. n the **Create Service** window, click **Save and Close**.
5. On the **Services** tab, select "NewTemplate" in Debug status, then click **Create Request**.
6. On the **Submit Service Request** window **Settings** pane, enter the following information.

Parameter	Description	Values
Volume Settings		

Parameter	Description	Values
Configuration Manager Connection	Configuration Manager connection	Select the Configuration Manager Connection from the table.
Host Settings		
Number of Hosts	The number of hosts for which to allocate volumes.	Single
Host Name	Host name	Enter the host name.
WWN Settings	WWN settings	Click the + icon and enter the information.
Task Settings		
Task Name	Name of the task	NewTemplate
Description	Brief task description	A task to generate an email for the service.
Schedule Type	Time the task runs	Immediate

7. After you finish entering the required values, click **Submit**, then click **OK** in the **Submit Service** confirmation dialog box.
8. On the **Tasks** tab, in Debug view, select the "New Template" task, then click **Show Details** to view the task summary, details, result, log, and notes.
9. After testing the service template to verify that the new service template is functioning correctly, you can return to the Service Builder **Edit** window, then click the **Release** tab to make the template available for users to submit.

Chapter 5: Creating a new plug-in

You can make a copy of an existing plug-in and then modify it, or you can create a new plug-in that runs commands, runs scripts, and completes the standard tasks for a service template.

Each plug-in has a purpose and the use and sequence in a service template is an important part of creating a service template.

Plug-in creation workflow

You manage and create new plug-ins from the Service Builder **Home** window Custom Plug-in Actions menu.

Use the Custom Plug-in Actions menu to edit, copy, or delete an existing plug-in.

When creating a new plug-in, you must complete the phases described in the following workflow:

Phase 1 - Preparation

1. Decide the purpose of the plug-in. Consider the steps for automating the process and determine if one or more plug-ins are required and if you can modify an existing plug-in or you must create a new plug-in.
2. Prepare to create the plug-in. This involves defining the plug-in and the associated icon file, preparing the required commands or scripts to run tasks, and preparing resource files.

Phase 2 - Creation

1. Create a new plug-in or copy and modify an existing plug-in. The plug-in is now in the Developing state.
2. Enter the standard information and set input and output properties.
3. Set the remote commands and environment variables as required.

Phase 3 - Testing

1. While developing a service template, drag the plug-in to the template flow.
2. Build the service template for testing.
3. Complete testing.
4. Debug the plug-in.
5. Rebuild and retest the service template until the plug-in runs successfully in the template.

Phase 4 - Releasing

Release the service template. The status of the service template and the related plug-ins changes to Released.

Creating a plug-in

You can create a plug-in based on one of the standard plug-ins, or you can create a new plug-in that runs the commands or scripts for a step in a service template.

Note: You cannot duplicate a plug-in with the same Plug-in key name, Vendor ID, and Version Number.

Before you begin

- Determine how to handle the script input properties and the method to deliver an output property.
- If running a script by file, create the script files.

Procedure

1. From the Service Builder **Home** window, access the **Custom Plug-in Actions** menu and choose **Copy** to use one of the existing plug-ins as a model, or choose **Create** to create a new plug-in from scratch.
   The **Copy Custom Plug-in** or **Create Custom Plug-in** dialog box opens depending on your choice of creation methods.
2. From the **General** tab, enter the basic information for the plug-in.
3. From the **Property** tab, click the input or output property icons and then add the input and output properties associated with the plug-in. You can add multiple input properties and drag and drop within the section to change the order of the input properties.
4. From the **Remote Command** tab, click **Add Platform** to select an Operating System. Specify the Credential Type and other plug-in details.
5. If remote commands or the script need environment variables, click **Details** and then click **Add** to enter the environment variable name and value.
6. Finish adding variables, then click **Save**.

Result

A new plug-in is created in the Developing state, which can be accessed from the Service Builder **Edit** window Flow tab.

Create/Edit Custom Plug-in dialog box

When you create a new plug-in, you provide the details from the General, Property, or Remote Command tabs in the **Create/Edit Custom Plug-in** dialog box.

General Tab

From the General tab, you can view and edit general details for the selected service template.

The following table describes the options available from the General tab.

When you enter information in a dialog box, if the information is incorrect, errors include a description of the problem at the right side of the box.

Field	Subfield	Description
Key Name *	-	Key name for the plug-in (up to 64 characters). The combination of the Key Name and Vendor ID cannot exceed 115 characters.
Version *	-	Version number of the plug-in.
Vendor ID *	-	Vendor ID of the plug-in (up to 64 characters). The combination of the ID and Vendor ID cannot exceed 115 characters.
Display Name	-	Name of the plug-in (up to 64 characters).
Vendor Name	-	Vendor name of the plug-in (up to 64 characters).
Description	-	Brief description of the plug-in (up to 1,024 characters).
Tags	-	Assigned Tag category for the plug-in. Click the Plus Sign to add tag categories.
Icon	-	Specifies the icon graphic (a 48 pixels × 48 pixels PNG file) associated with the plug-in. To change the default image provided, click Change and enter the graphic file name. You can revert to the original icon anytime by choosing Restore Default Icon.
Fields marked with an asterisk (*) are required.		

Property Tab

From the Property tab, you can search for and view the input and output properties for a service template. You can search for a property with the text search box. Click input or output (next to search box), to switch between the lists for input and output properties. You can also edit a property by clicking the pencil icon.

The following table describes the options available from the Property tab.

Field	Subfield	Description
Key Name	-	Displays the key associated with the property.
Display Name	-	Displays the name associated with the property.
Description	-	Gives a description for the property.

Field	Subfield	Description
Required	-	Indicates whether the property is required (true) or (false).
Default Value	-	Specifies default values if required.

Remote Command Tab

From the Remote Command tab, you can set up remote commands that run during the processing of a service template.

To add a remote command, click Add Platform and then provide the details for the OS environment.

The following table describes the options available from the Remote Command tab.

When you enter information in a dialog box, if the information is incorrect, errors include a description of the problem at the right side of the box.

Field	Subfield	Description
Credential Type	-	Specifies the credential type required for the plug-in. To use the credential information in the agentless remote connections view under the Ops Center Automator Administration tab when the service runs, select Shared agentless setting. Shared agentless setting is the default Credential Type. The following reserved plug-in property is automatically set for the shared agentless setting credential type: ▪ plugin.destinationHost Enter the target of an action with an IPv4 address, IPv6 address, or host name (up to 256 characters). To use the credential information as an input property, select Service input property.

Field	Subfield	Description
		The following reserved plug-in properties are automatically set for the Service input property credential type: ▪ plugin.destinationHost Enter the target of an action with an IPv4 address, IPv6 address, or host name (up to 256 characters). If the destination host is the Ops Center Automator Server (localhost), the user ID and the password are not necessary. ▪ plugin.account Enter the user ID for logging on to the target host (up to 256 characters). ▪ plugin.password Enter the password for logging io to the target host (up to 256 characters). ▪ plugin.suPassword Enter the password of the root account used for logging on to a target host in a Linux OS environment (up to 245 characters). If the target host is running the Windows OS, this property is ignored.
Windows Options	Run as system account	Runs the command using the system account.
Linux/UNIX Options	Execute with root privileges	Runs with root privileges.
	Character Set Auto Judgment	Character Set Auto Judgment applies to the Linux OS. If enabled, the script runs using the default locale of the user. If disabled, the script runs with the LC_ALL=C locale. The default is enabled.
Add Platform	-	Specifies the platform for the remote command (Windows or Linux platform).
Import Settings	-	Specifies to obtain the settings from another OS.
Execution Method:	-	Select Script name if based on a script, or CLI Command if using a stored command.

Field	Subfield	Description
CLI Command: *	-	Enter the CLI command (up to 8,192 characters) for the remote command. This is required if you choose Script name or CLI Command for the Execution Method. Use Insert All Input Properties to insert all input properties defined for the plug-in to the command.
Type in:	-	Select By attachment if uploading a script file, or Type in to directly enter the script information. Use the following file format for a plug-in that runs a script: *name-of-plug-in.extension*.
File*	-	Click Browse to specify the file that contains the command or script.
Mapping Definition of Output Properties	-	Specify the mapping definition of the output property for a command or script. Select and highlight a property in the list and then click Edit to access the **Edit Output Filter** dialog box where you can specify an output filter that controls the data that is passed to the output property.
Execution Directory	-	Specify the folder in which the command or script runs.
Environment Variables	-	Select environment variables for the command or script as required. The **Create/Edit Environment Variable** dialog box appears for you to enter the variable information.
Fields marked with an asterisk (*) are required.		

About plug-in properties

You define input and output properties for plug-ins to specify the parameters required when running a task and processing the results.

You must map the input and output properties of the components for each step with the input and output properties and variables of the service template. The input properties of the components used in a service template must be associated with the input properties or variables, or output properties of other component steps associated with a service template. The service template input properties store the input values that are required to run a service. The service template output properties store the results of running the service.

The value of the plug-in input property can be the value that you previously set for the property, or directly set for the service template. The input property can also be a variable. You can apply service share properties to the input properties.

The output property content depends on the type of component. You can store the result of running a step as output properties. Variables temporarily hold the values that are passed between components.

The input and output properties of the components are set from the Flow tab of the Service Builder **Edit** window. The input and output properties and variables for the service template are set in the Property tab of the Service Builder **Edit** window.

Input and output properties can store a maximum of 1,024 characters unless you specify "composite" as the data type for the Input/Output property. If you specify a value that is more than 1,024 characters without specifying the composite data type, the first 1,024 characters are stored as the property value and the remainder are discarded. If you reference the value of a property key in the format `?dna_property-key?`, the referenced value is truncated if it is more than 1,024 characters.

A plug-in property with a property key and purpose, which are determined in advance, is called a reserved plug-in property. These properties specify credential information and the target hosts of remote commands.

Add plug-in input properties

Input properties store the values that plug-ins need when running, such as the arguments for remote commands or the target host of the action. Create and define input properties from the **Create Custom Plug-in** or **Edit Custom Plug-in** dialog boxes.

From the Service Builder **Home** window, choose either Create or Edit from the Custom Plug-in Actions menu.

From the Property tab of the **Edit Custom Plug-in** dialog box, make sure that the input listing is selected (by clicking the input property icon) and then click Add to enter the required input properties. You can add multiple input properties and drag and drop from in the section to change the order of the input properties.

Continue to enter the properties and remote commands of the plug-in.

Specify/Edit Input Property for Custom Plug-in dialog box

The following table describes the **Specify/Edit Input Property for Custom Plug-in** dialog box fields, subfields, and field groups.

When you enter information in a dialog box, if the information is not valid, errors include a description of the problem at the right side of the box.

Field Group	Field	Description
Definitions	Key: *	Input property key name.
	Display Name: *	Name of the input property.
	Description:	Description of the input property.

Field Group	Field	Description
	Visibility:	Choose whether the input property is visible on both the Edit and Submit windows [Edit and Submit Window], or just in the Edit window [Edit Window Only].
	Display Settings:	Specifies the display setting for the input property. The choices are: ▪ Editable ▪ Read only ▪ Hide
	Required:	Specifies that the property is required when this check box is checked.
	Data Type:	Select a data type of the property: string, boolean, integer, double, date, password, composite. Various options are presented for specifying restrictions on the data entry, depending on the options you chose. When dealing with arrays, you can verify the Array Type option for the data type to be treated as an array. A set of the properties of the same type (Number of elements is a variable) can be handled as a single property to make data mapping easier, especially when passing data between a service and the plug-ins. Verifying the File Reference check box specifies that the value of the property is expressed as the file path. The value of the property is automatically stored in a file, and you can retrieve the path of the file as an input property. For example, you can use a file path in the command line instead of a direct value.
	Content Type:	Select the content type: ▪ application/json ▪ application/javascript ▪ application/xml ▪ text/html ▪ text/plain

Field Group	Field	Description
		▪ text/csv ▪ application/octet-stream
	Domain Type:	Enter the domain type from the list or add a new domain type by clicking the Plus Sign and entering the details from the Create Domain Type Definition dialog box. This option is available when you choose composite for the Data Type and application/json for the Content Type.
Restrictions	Minimum Value/Length:	Specifies the minimum value for an integer and double. If the data type is string or password, then enter the minimum length of the property. If the data type is date, then enter the earliest date.
	Maximum Value/Length:	Specifies the maximum value for an integer and double. If the data type is string or password, then enter the maximum length of the property. If the data type is date, then enter the current date.
	Restricted Character:	If the data type is string or password, then enter the allowed characters by using a regular expression. Example: ^[0-9a-zA-Z\.\-]*$
	Minimum Array Length:	Specifies the minimum length of array elements.
	Maximum Array Length:	Specifies the maximum length of array elements.
	Validation Script:	Validates the property based on the associated javascript code.
Value and Presentation	Presentation:	Specifies options that determine how the property selection is presented, depending on the selected Data Type.

Field Group	Field	Description
	Default Value:	Specifies the default value for the property. The values that can be specified differ depending on the data type. When specifying the Array Type option, the default value must be written as a comma-separated string value surrounded by brackets. Example: ["1","2","3"]
	Data Source:	Specifies the data as Static or Dynamic and derived from an external resource provider.
	Specify List Items:	Specifies the "Specify List Items" when the data source for the property is derived statically (when choosing the Static option for the Data Source).
	External Resource:	Specifies the external resource provider when the data source for the property is derived dynamically (when choosing the Dynamic option for the Data Source. In the list, you can also add, edit, upload, or delete for the external resource provider.
	Extra Path:	Specifies the extra path portion of the request URL. Leave it empty if it is not required. The extra path is the path that follows the external resource provider ID in a URL as follows: /Automation/v1/objects/ExternalResources/*<external resource provider ID>*/*<extra path>*?*<query parameters>*
	Query Param:	Specifies the query parameter for the external resource provider. The `serviceID` and `serviceTemplateID` parameters are added automatically. You can specify {$ref:*keyName*} to embed the property value of other properties. For a JSON value, you can specify {$ref:*keyName#json path*}.
	Name Field:	Specifies the field name of the object collection to use the label of the selection. If omitted, uses the name field.
	Value Field:	Specifies the field name of the object collection to use the label of the selection. If omitted, uses the instance ID field.

Field Group	Field	Description
	Show If:	Show the property entry if the conditions are met.
	Enable If:	Enable the property if the conditions are met.
Fields marked with an asterisk (*) are required.		

Adding plug-in output properties

The output properties store the values that plug-ins need when running, for example, the arguments for remote commands or the target host of the action. Output properties are created and defined from the **Create Custom Plug-in** or **Edit Custom Plug-in** dialog boxes.

From the Service Builder **Home** window, choose either Create or Edit from the Custom Plug-in Actions menu.

From the **Edit Custom Plug-in** dialog box Property tab, verify that the output listing is selected (by clicking the output property icon) and then click Add to enter the required output properties. You can add multiple output properties and drag and drop from in the section to change the order of the output properties.

Continue to enter the input and output properties and remote commands for the plug-in.

Specify/Edit Output Property for Custom Plug-in dialog box

You can add or modify output properties for a service template.

The following table describes the Specify/Edit Output Property for Custom Plug-in dialog box fields, subfields, and field groups.

When you enter information in a dialog box, if the information is not valid, errors include a description of the problem at the right side of the box.

Field Group	Field	Description
Definitions	Key *	Output property key.
	Display Name:*	Name of the output property.
	Description:	Description of the output property.
	Display/Hide:	Specifies the display setting for the output property. The choices are: ▪ Display ▪ Hide
	Data Type:	Select a data type of the property: string, boolean, integer, double, date, password, composite. Various option are presented for specifying restrictions on the data entry depending on which of the options are chosen. When dealing with arrays, you can verify the Array Type option for the data type treated to be as an array. In this way, a set of the properties of the same type (Number of elements is variable) can be handled as a single property thus making data mapping easier, especially when passing data between a service and the plug-ins. Verifying the File Reference check box specifies that the value of the property is expressed as the file path. You can use the output value in the next step through a file. For example, when you run the following command: **<command> > "?dna_output?"** in the plug-in, and "?dna_output?" is verified in the File Reference, you can store the output value to the file whose path is "?dna_output?" and use next step input value.
	Content Type:	Select the content type: ▪ application/json ▪ application/javascript ▪ application/xml ▪ text/html

Field Group	Field	Description
		▪ text/plain ▪ text/csv ▪ application/octet-stream
	DomainType:	Select the domain type from the list or add a new domain type by clicking the Plus Sign (+) and entering the details from the **Create Domain Type Definition** dialog box. This option is available when you choose "composite" for the Data Type and "application/json" for the Content Type.
Value and Presentation	Presentation:	Specifies how the property selection is presented, depending on the Data Type that you selected.
	Show If:	Show if the conditions are met.
	Enable If:	Enable if the conditions are met.
Fields marked with an asterisk (*) are required.		

Setting remote commands in plug-ins

Plug-ins use remote commands to pass an input property to a script or command. Remote commands are also used to filter an output property from the standard result output. Set an output filter to store the value you want from the standard output. A plug-in needs one or more remote commands. Environment variables are set through remote commands.

Procedure

1. From the Service Builder **Home** window, choose either **Create** or **Edit** option from the **Custom Plug-in Actions** menu. If you are editing an existing plug-in, select the plug-in from the card or table view, click **Edit**.
   The **Create/Edit Custom Plug-in** dialog box appears.
2. From the **Remote Command** tab, click **Add Platform** to select the operating environment.
   A choice of operating platforms is provided.
3. Choose the platform from the list and then enter the relevant details.
   Options are provided depending on the platform you have chosen.
4. Enter all of the details for the remote command and then click **Save**.
   The remote command is created for the selected plug-in.

Setting environment variables

You can set environment variables when creating or editing a plug-in from the Remote Command tab of the **Create/Edit Custom Plug-in** dialog box.

Procedure

1. From the Service Builder **Home** window, choose the **Create** or **Edit** option from the **Custom Plug-in Actions** menu.
   After you have entered the details for a new plug-in, or have selected an existing plug-in to edit from the **Custom Plug-in List** dialog box, the **Create/Edit Custom Plug-in** dialog box appears.
2. From the **Details** section of the **Remote Command** tab, under the **Environment Variables** area, click **Add**.
   The **Create Environment Variable** dialog box appears.
3. Enter the relevant attributes (name and value) for the environment variable, then click **OK**.
   The defined environment variable and its associated value is shown. You can continue to add more environment variables or edit existing ones.

Create/Edit Environment Variable dialog box

You can enter or modify environment variables for a remote command from the Details section of the Remote Command tab that you accessed from the **Create/Edit Custom Plug-in** dialog box.

The following table describes the Create/Edit Environment Variable dialog box fields, subfields, and field groups.

When you enter information in a dialog box, if the information is not valid, errors include a description of the problem at the right side of the box.

Field	Subfield	Description
Name: *	-	Name of the environment variable.
Value:	-	Value of the variable.
Fields marked with an asterisk (*) are required.		

Adding output filters

Standard output and standard error output are stored in the output property. Therefore, the best practice is to set filters on the standard output results to obtain the value you want by using regular expressions in the remote commands.

Example

To obtain the disk ID from the standard output property results, complete the following steps:

1. Set a filter on the output property:

   `diskid:n` is the output method as the result of a script

   `diskid:(.+)` is the regular expression to filter the output property

2. Create a script to pass the results.
3. Use a remote command to set an output filter where:

   `diskid:(.+)` is the output filter

   `diskid:1` is the standard output string

   `blank` is the value of the output property

Running the script gives the following results:

- `diskid:1` is the standard output string
- *`1`* is the value of the output property

You specify output filters from the Remote Command tab that is accessed from the **Copy/ Edit Custom Plug-in** dialog box.

To set an output filter:

From the **Mapping Definition of Output Properties** area, select and highlight the line of the output property, then click the edit (pencil) icon. Enter the details in the **Edit Output Filter** dialog box.

Continue to set environment variables as required.

Note: When specifying multiple groups in a regular expression, only values that match the first group are stored in the output property. In addition, If the regular expression applies to multiple value ranges, only the first range of values is stored in the output property.

Note: Regular expressions are evaluated in the multiline mode. To eliminate unwanted characters (including the line feed), you must write the regular expression specifying the single-line mode. The single-line mode is described as "(?s)" in the regular expression.

The cutout by the regular expression is the part grouped by the parentheses. The first group becomes the target of the cutout if multiple groups exist. Here are some examples of cutting out the standard output using a regular expression.

Example of extracting the string from a single line:

```
Standard output
server:sv001
CPU - 89%
Memory - 77%

Regular expression
server:(.*)

Result of cutout
sv001
```

Example of extracting the string with multiple lines - 1:

```
Standard output
server:sv001
CPU - 89%
Memory - 77%

Regular expression
server:(?s)(.*)

Result of cutout
sv001
CPU - 89%
Memory - 77%
```

Example of extracting the string with multiple lines - 2:

```
Standard output
server:sv001
CPU - 89%
Memory - 77%

Regular expression
server:(?s)(.*)\sMemory

Result of cutout
```

```
sv001
CPU - 89%
```

Edit Output Filter dialog box

You can specify an output filter using a regular expression to control the data that is processed by the output property. If the output filter is empty, standard output and standard error output is stored directly in the output property.

The following table describes the **Edit Output Filter** dialog box fields, subfields, and field groups.

When you enter information in a dialog box, if the information is not valid, errors include a description of the problem at the right side of the box.

Field	Subfield	Description
Output Filter:	-	Enter a regular expression to filter the data stored by the output property.
Verification of the Output Filter	-	Performs the processing and verifies that the filter is processing the data in the way you intended. The standard output details and results are shown.

Creating a conditional branch using the branching plug-ins

You can create a conditional branch so that a step within a service template is run only when a particular condition is met.

A conditional branch is useful for running a step based on a condition that occurs during the processing of a previous step.

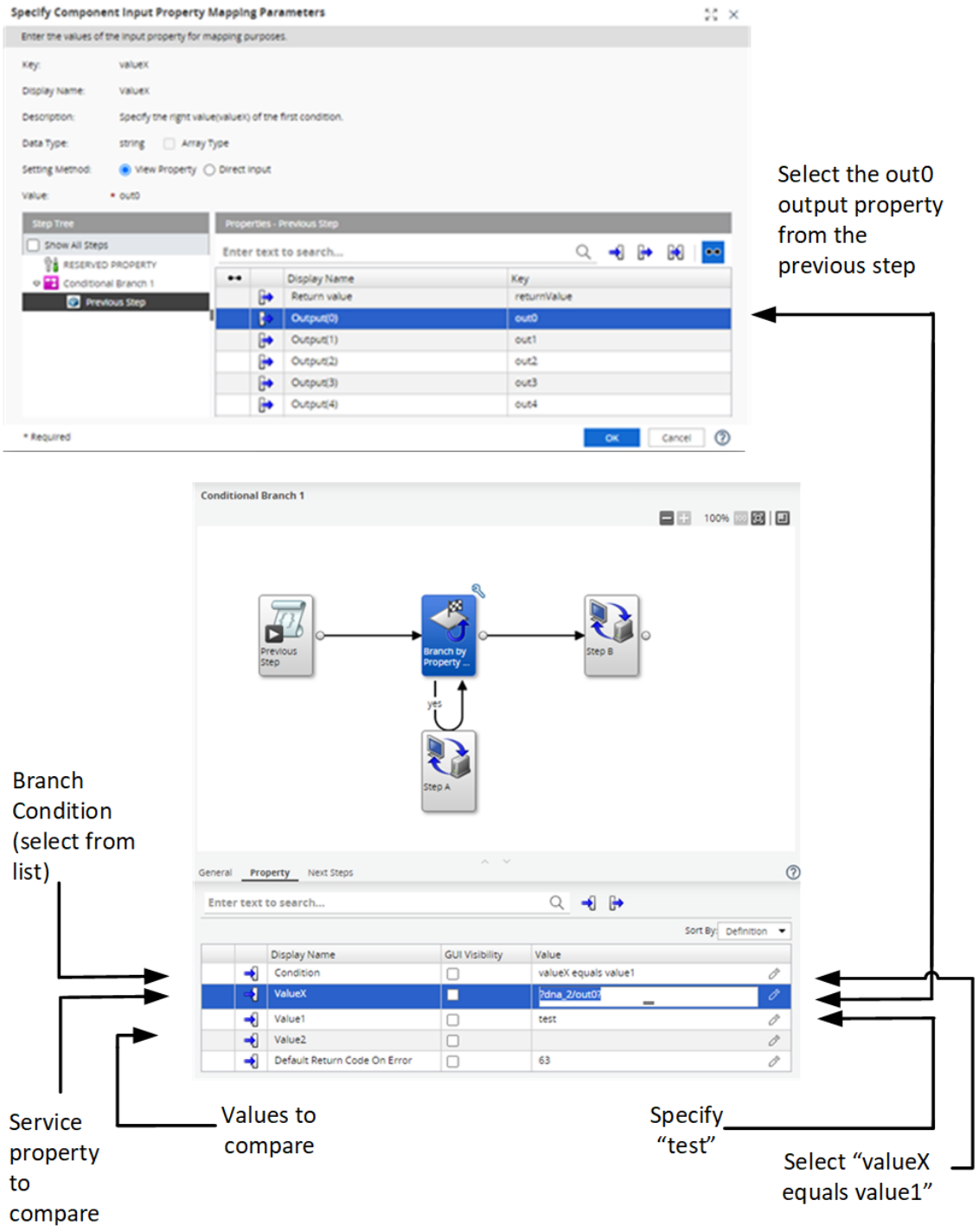
The following plug-ins are provided for creating a conditional branch:

- Branch by Property Value Plug-in: Compares a service property with a specified value (or if necessary, can also be compared with a variable or step property) and then acts accordingly.
- Branch by ReturnCode Plug-in: Compares the Return Code generated by a previous step with a value.

To create a conditional branch:

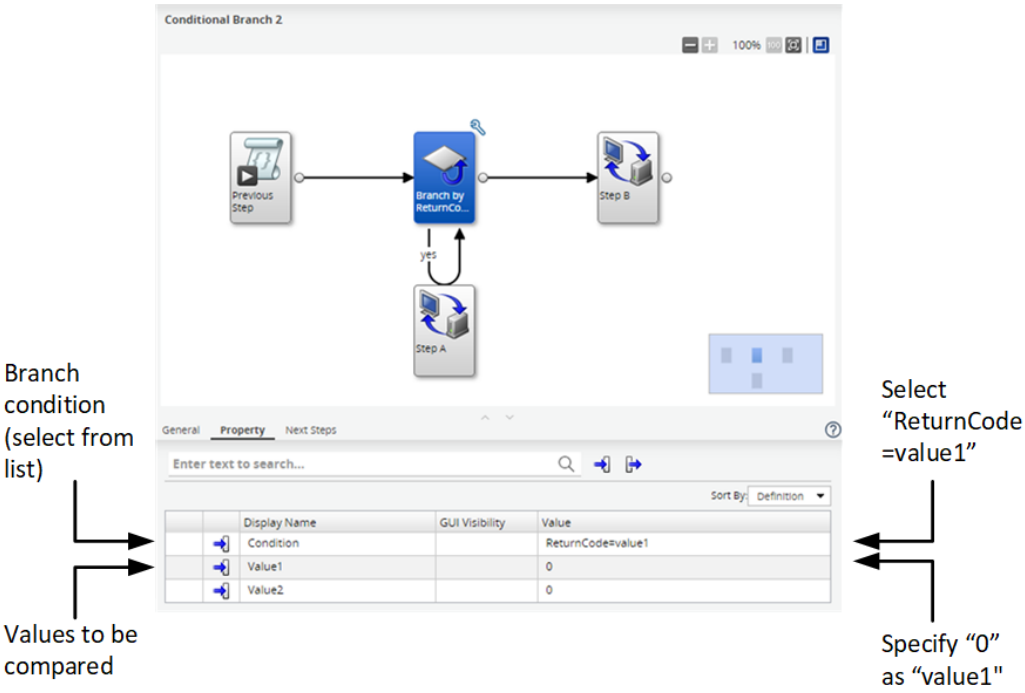
1. Insert the branching plug-in at the point in the flow after the step where you want the branch to occur.
2. Position the step to run, if the condition is met, beneath the branch and the position of the next steps.
3. Draw connectors between the steps to define the flow.
4. Set the input and output values to define the condition and specify the values.

The following figure shows an example of using the Branch by Property Value Plug-in to set up a conditional branch when a step property generated by a previous step equals the specified value.



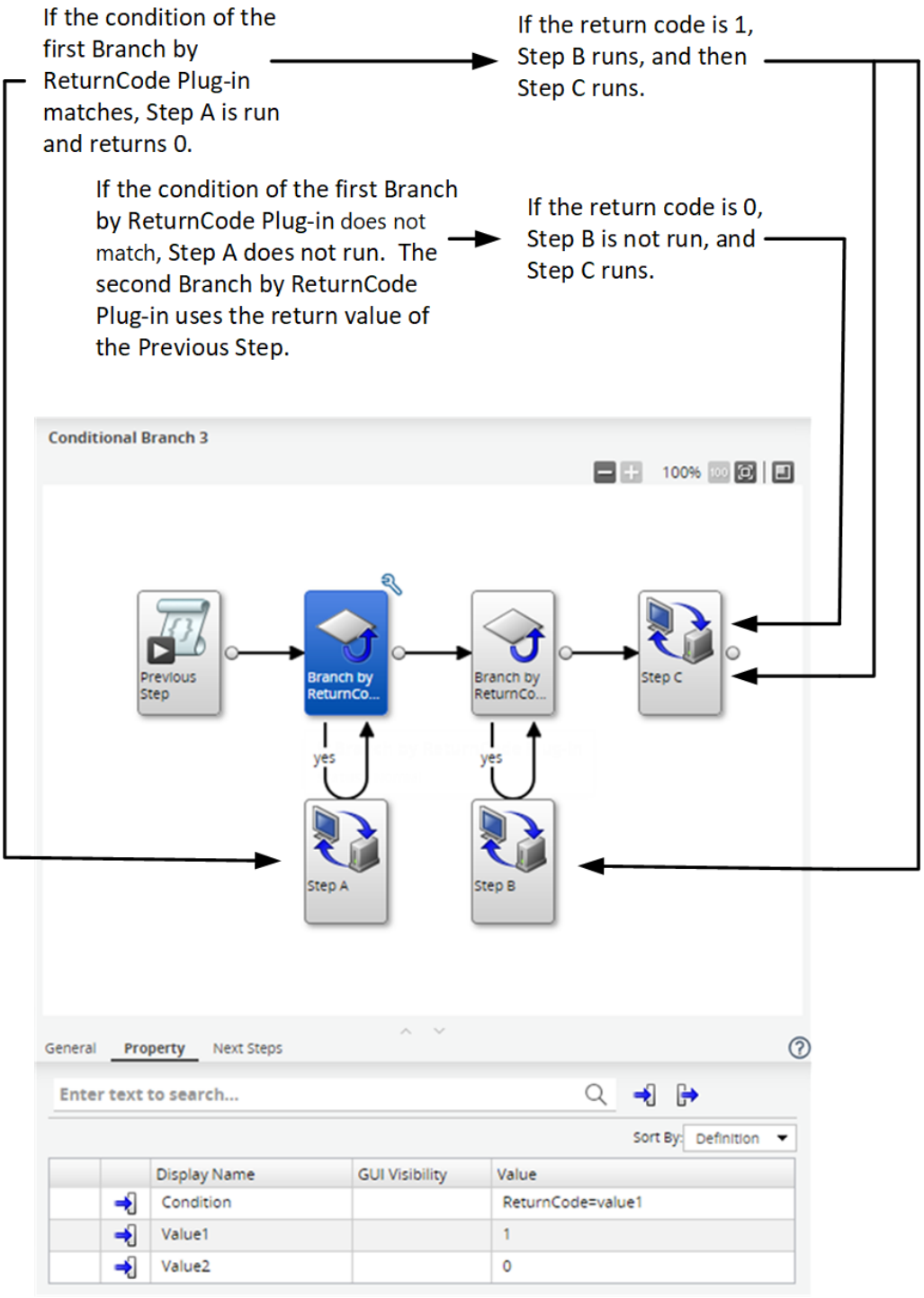
In this example, "Step A," is run only when "out0" from the previous step equals "test."

The following figure shows an example of using the Branch by ReturnCode Plug-in to set up a conditional branch when a return code generated by a previous step meets a specified criteria.



In this example, "Step A" is run only when the return code from the previous step equal "0."

To create a more complex conditional branching, you can use multiple Branch by Returncode plug-ins so that one step is run when a condition is met and another step is run when the condition is not met.



Generating an email

During the processing of a service template, you can generate an email with a specified notification or the contents of some output by using plug-ins.

You can use the Email Notification Plug-in to send an email to specified recipients notifying them when a specific condition occurs and that includes information that has been output from a task or process. For example, you can send the result of a LUN path configuration associated with the Allocate Volumes with Smart Provisioning Service to a specified email address. When using this plug-in, you specify the address for the recipient, a subject line, and the body of the email which can consist of a predefined message or the output from a previous step. You can also specify the encoding for the formatting of the email.

Details about the email sender are defined in the SMTP server settings. Before using the Email Notification Plug-in to send an email, you must specify the correct email details from the Email settings, accessed through the System Settings, available from the Administration tab.

To generate an email:

Procedure

1. Insert the **Email Notification Plug-in step** at the point in the flow after the step that has the information for the email notification.
2. Draw connectors between the steps to define the flow of execution.
3. Click the **Email Notification Plug-in step** and enter the values for the input properties specifying the recipient's address, encoding, and subject.
4. In the Body field of the Email Notification Plug-in, enter the key for the output property in the plug-in step that is generating the email content.

   In some cases, you might want to send output that is not in the default JSON format that you must first convert before sending the email. To accomplish this, you can use the JavaScript Plug-in. This plug-in treats JSON input data as an object that can be converted through a JavaScript script to the required format.

Chapter 6: Building, debugging and releasing

After specifying the steps and establishing the flow for a new or modified service template, you must access the Debug tab to generate a build. After a build is successfully completed, you can use the built-in debugger to run through the steps and correct any problems with the flow or property mapping. When the service template is functioning properly, you can release it to the operating environment where a user can use it to create a service.

Debug and release workflow

After creating or modifying a service template, you can build and then debug the service according to the following phases:

Phase 1 - Preparing

Before building the service template, make sure that the service template details are specified, that the plug-in steps are arranged in the correct flow order, and that values for the input and output properties are properly defined.

Phase 2 - Building

1. Start building the service template by clicking on the Debug tab.
2. Observe the information returned by the Build / Release dialog box to see if any errors are generated.
3. If there are errors, hover the cursor over the error message to see details pertaining to the error.
4. Make whatever corrections are necessary and continue to rebuild the service template until it completes building successfully.

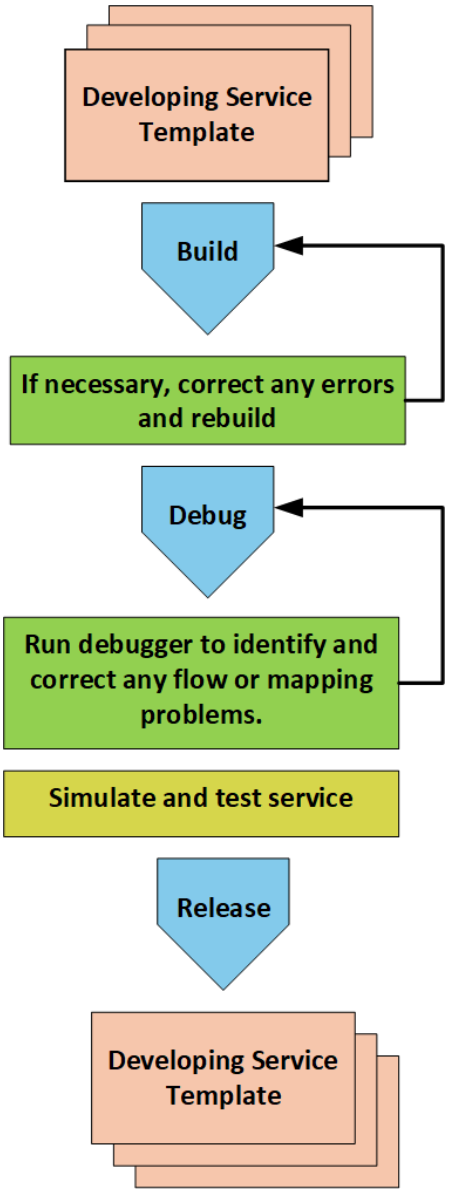
Phase 3 - Debugging

1. After a service template has been successfully built, you can perform any necessary debugging.
2. Correct any errors and continue re-building the service template until all errors are resolved.
3. After all problems have been resolved, conduct a test by adding and executing the service in the development environment.

Phase 4 - Releasing

1. When the service template is functioning properly and there are no further problems, you can release the service template. A service template must be in Released status to submit the service template to the operating environment.
2. Create a service from the released service template.
3. Verify the execution results of the task. If any problems are identified, amend the affected service template or plug-in and repeat the debug process.

The following figure shows the typical steps you follow when debugging, testing and releasing a service template.



Building a service template

After a service template is created and is in the Developing status, the next step is to build and debug the service template and its related plug-ins.

Here are the steps for initiating the building and debugging of a service template:

Before you begin

A service template in the Developing state must exist.

Procedure

1. From the Service Builder **Edit** window, click the **Debug** tab.
   After confirming your intention to perform the debug, the **Build / Release Results** dialog box appears and the service template undergoes the build process. If any errors are generated, error icons and links are visible and warning icons appear when you have failed to enter a required property.
2. Make the necessary corrections where indications appear or adjustments to the process as needed.
3. Repeat the build process until the service template is free of errors.

Result

After building a service template, the following processes occur:

- Services added from the debug version of the service template are deleted, and tasks run from the service are archived.
- The debug task run from the debug version of the service template is deleted.
- The debug version of the imported service template is deleted and then re-imported.

After the build process completes successfully, the **Perform Debugging** dialog box appears where you can verify the flow of execution and make any adjustments that are necessary before releasing the service template.

Next steps

- Access the debugger to verify the flow of steps and to make any required modifications.
- Create services and tasks based on the debug configuration of the service template.

  For more information on creating services, see the *Hitachi Ops Center Automator User Guide*.
- Provided the service template works properly and has passed the build process, proceed to release the service template.

Build / Release Result dialog box

You initially verify the results when building and releasing a service template from the **Build / Release Result** dialog box.

The following table describes the **Build / Release Result** dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

Field	Subfield	Description
Results Summary	Build / Release Result	Displays the results of the service template build.

Field	Subfield	Description
	Status	Displays the current status for the service template.
Status Details	Type	Indicates the type of status message.
	Message ID	Shows the message ID. Use the following file format for a plug-in that runs a script: *name-of-plug-in.extension*.
	Message	Shows the content of the status message.

After the service template builds successfully, you can start the debugging process from the **Perform Debugging** dialog box.

Running the debugger

After successfully building a service template, you can run the built-in debugger by accessing the **Perform Debugging** dialog box and then clicking OK. Before accessing the debug interface, you must specify the details for the debug service and tasks in the **Perform Debugging** dialog box. You can also verify and edit the service properties or make changes from the **Create Service** or **Create Request** windows.

You can use the built-in debugger to make sure the plug-ins and flow of a service template are working as intended. During a debugging session, you can:

- Control when to run steps in the flow of a service template to isolate and correct any problems.
- Run and manage debug tasks while verifying the flow transitions at all hierarchical levels.
- Confirm that property mapping is set correctly and that the conditions for running subsequent steps flow as intended.
- Modify input and output properties for currently running tasks.
- Set breakpoints to enable the processing to start from or end before a specified step in the flow.
- Skip the processing of specified plug-ins (script/command, repeat, wait for user response) so that it appears to the next step as though the process of the plug-in has run, which allows the flow to continue according to the condition of the next step.
- Display the results of a repeated-execution flow (for each execution time).
- Consult a running log of running tasks.

- Edit entries in the **Create/Edit Service** and **Submit Service Request** windows to simulate service template processing.
- If you detect an problem with a plug-in, you can assign an arbitrary property value or return value to the plug-in and run the plug-in again. This allows you to see how the plug-in processing and flow transitions are effected when you change a specific property value or return values.

You can debug a service template multiple times.

When you debug a service template, Ops Center Automator creates a debug service and debug task.

Debug service

A debug service is a service that is generated and run when you debug a service template. One debug service is created per service template. When you debug a service template that has already been through a debug process, Ops Center Automator deletes the existing debug service and creates a new one. The debug services appear in the Service column in the Tasks - Debug view, but do not appear in the **Services** window.

Debug task

A debug task is a task generated for a debug service when debugging a service template. When you debug a service template that has already been through a debug process, Ops Center Automator deletes the existing debug task and creates a new one. Debug tasks appear in the service template Debugging view and Tasks - Debug view. Only users assigned the Admin or Develop role can view and work with debug tasks. Debug tasks do not appear in the task summary.

Note: You cannot edit service template or plug-in definitions from the debugger view. Instead, when you detect a defect during debugging, you must stop debugging and return to the Service Builder **Edit** window to make the required correction.

Limitations on concurrent debugging procedures and associated tasks

A maximum of one debug service and one debug task can be generated for the same service template, and the same service template cannot be debugged by multiple users at the same time. The same service template cannot be edited by multiple users at the same time (because the last save action for the template takes precedence, which means that multiple problems cannot be fixed at the same time).

If a debug service and a debug task already exist for a service template that is being debugged, when you run the build or release process, the debug service and debug task are deleted automatically, and a new debug service and debug task are created when debugging re-starts.

After the debug task finishes, if you start debugging again without first closing the Debugger view, the created debug service and the finished debug task are deleted automatically and a new debug service and debug task are generated.

Before you begin

A Developing version of the service template to debug must have successfully completed the build process.

Follow these steps to start a debugging session:

Procedure

1. Enter the details from the **Perform Debugging** dialog box.
2. For the **Service Name** field, accept the original name appended with [DEBUG] or specify another name of your choosing.
3. Verify and, if necessary, modify the details for the other service and task related fields
4. For the **Task Log Level**, specify the level of details to store in the task log file.
5. If necessary, you can supply the values that are normally specified by the user who is configuring and submitting the service by accessing the **Edit** menu and selecting either **From Create Service** window or **From Create Request** window.
6. Click **OK** to access the debugger user interface.

Result

The debug interface opens where you can begin the debugging process.

Perform Debugging dialog box

After successfully building a service template, you can specify the details for the debugging session in the **Perform Debugging** dialog box.

The following table describes the **Perform Debugging** dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is incorrect, errors that include a description of the problem appear at the right side of the box.

Field	Subfield	Description
Service Name: *	-	Specifies the name assigned to the service template. The debug version has "[DEBUG] appended to the name.
Tags:	-	Specifies the tag groups with which the service is associated.
Task Name:	-	Specifies the assigned task name. The debug version has "[DEBUG] appended to the name.
Task Description:	-	Gives an optional short description for the task.

Field	Subfield	Description
Service Group:*	-	Specifies the service group with which the task is associated. Because the target device operated by the debug task affects the agentless connection-destination definition, you must specify a service group.
Task Log Level:	-	Specifies the level of information that is output to the task log during the debug process. ▪ Severe ▪ Information ▪ Fine ▪ Finer ▪ Debug (default)
Properties The following details are provided for the list of properties: ▪ Display Name ▪ Key Name ▪ Value ▪ Description ▪ Scope	Edit	Allows you to edit the service from the Modify and Submit user perspective. ▪ From Create Service Window ▪ From Create Request Window
	Restore Default	Restores the default values of properties when building the service template.
	Import	Imports a properties file. This can be useful for importing a properties file with known settings from another service template or that was saved at some point for the current service template that you are debugging.
	Export	Exports a properties file. This can be useful for saving the properties for a service template to a file before making changes, which allows you to import the original property settings if necessary at some point during the debugging process.

Editing service and request entries while debugging

Besides specifying the service and task details for the debugging from the Perform Debugging dialog box, you can also examine and edit the service properties to supply the config and submit values from the Create Service or Create Request windows.

The properties section shows a listing of service properties implemented for the service template. It shows the Display Name, Key Name, Value, Description and Scope.

Before proceeding to the debugger interface, you often must first specify the values that normally are supplied by the Modify and Submit users when running a service. The following messages indicate that you must supply values for the specified properties before continuing with the debugging session:

```
Information is not complete. Please edit properties from [Create
Service Window]
```

```
Information is not complete. Please edit properties from [Create
Request Window]
```

To edit the properties from either the Create Service or Create Request windows, follow these steps:

1. From the Properties section of the Perform Debugging dialog box, click Edit.
2. Choose **From Create Service Window** or **From Create Request Window** depending on which interface you want to edit.
3. Click the category of settings (marked in red and with a warning (!) icon) and enter the missing values for all fields that are marked with a red asterisk (*), until the section turns blue indicating that all required fields have been filled in. Then click OK.
4. After specifying the service and task details for the debug session, and supplying any of the required values for the properties, click OK to access the debugger interface where you can begin debugging the service template.

While editing the service properties, you can also click Import to restore properties from a previous debugging session or Export to save the properties from the current session. In some cases, you might want to use this import capability to save time by loading all of the property values at one time, or possibly to restore property values that you previously exported. You can also click Restore Default to restore all default values.

Working with the debugger

After successfully building a service template in preparation for its release, and supplying the required service and task details from the Perform Debugging dialog box, you can use the Debugger interface to debug the service template.

The **Service Builder Debug** window has the following operational panes:

- Debugger: This area allows you to control debug processes associated with the currently selected step.
- Flow: Shows the placement and flow of steps associated with the service template.
- Flow Tree: Shows the hierarchical flow of steps.
- Task Log: Selecting this tab shows a list of task entries. You can update the list to see the current state of running tasks or download the list to a file for subsequent reference.
- Service Properties: Selecting this tab shows the Display Name of the service properties associated with the service template along with the assigned Key Name and Value.
- Break Points: Selecting this tab shows any currently set breakpoints along with their Flow Hierarchy and Display Name: You can remove all currently set breakpoints by clicking Remove All Break Points.

Debugger

The debugger has a convenient method for controlling how to run tasks for a debug process using the following options:

Icon	Option
	Input Response: Requests an input response for a waiting task.
	(Retry Debug): Tries the debugging again as follows: ▪ Retry Debug: Tries the debugging process again from the beginning of the service template. ▪ Retry the Task Starting from the Failed Step: Tries the task again, beginning from the last failed step. ▪ Retry the Task Starting from the Step after the Failed Step: Tries the task again immediately after the last failed step.
	(Resume Debug Operation): Resumes the debugging from the last failed step.
	(Interrupt Debug Operation): Interrupts the debugging at the current task.
	(Forcibly Stop Debug Operation): Forcibly stops the debugging at the current task.
	(Execution of Step Into): Runs the task to the next interruptible step.
	(Execution of Step Over): Runs the task to the first interruptible point in the next step.
	(Execution of Step Return): Runs the task to the first interruptible point in the upper hierarchy.
	(Set/Unset Breakpoint): Sets (or removes) a breakpoint that causes the running tasks to pause after the specified step.

Note: The system automatically generates a breakpoint when you choose the step-into, step-over, or step-return option to control how the tasks run during a debug session.

Debug Modes

- Run plugin in execution mode: Runs the plug-in in execution mode.
- Run plugin in dry-run mode: Runs the plug-in in dry-run mode.

Task Status

The status for a debug task can be one of the following:

Icon	Status
	Waiting: Indicates that the task is waiting to run.
	Interrupted: Indicates that the task action was interrupted by the step execution feature.
	In Progress: Indicates that the debug task is in progress.
	Waiting for Input: Indicates that the task is waiting for user input.
	In Progress (with Error): Indicates that the task detected a processing error.
	In Progress (Terminating): Indicates that the task received a stop operation or forcibly stop operation instruction, and is terminating the processing.
	Completed: Indicates that the task completed successfully.
	Failed: Indicates that the debug task failed.

Conditional Expressions in a Next Step Flow

When using conditional expressions in a next step flow, the line indicating the flow of execution shows the arrow conditional expression ( ) icon indicating its current status:

- Green Icon: TRUE
- Dark Gray Icon: FALSE
- Dark Gray Icon: NOT YET (not yet run)

By mousing over the arrow with the conditional expression icon attached, the arrow condition name, status, type, and description appear as a tool tip.

Step Information

Step information gives step details as follows:

- ID: Shows the name of the currently selected step.
- Display Name: Shows the display name of the step.
- Status: Shows the status of the debug action for the currently selected step.

Step Properties

The Step Properties view shows the input and output properties associated with a step along with the current values.

The following icons indicate the type of property:

Icon	Type
	Indicates an Output property
	Indicates an Input property

You can edit the property values directly by clicking the pencil icon that accesses the **Edit Step Property** dialog box where you can make any necessary changes.

While examining the step property values, you can click Edit and select either the **From Create Service** window or **From Create Request** window to specify the values for the properties normally supplied by the Modify and Submit users.

You can also click Import or Export to save or retrieve property values from a specified file.

The Task Log, Service Properties, and Break Point tabs provide additional useful information about the current debugging session.

Examining debug details

While debugging a service template, you can examine details for tasks that have run, current service properties, and breakpoints that were set.

During a debug session, you can access details from the following tabs:

Task Log

By clicking the Task Log tab, you can access a list of tasks that were run during the current debugging session. To update the task list with the latest task activity, click Refresh or you can click the Refresh Automatically check box so that the task list automatically updated.

To keep a record of the completed tasks for a debug session, click Download and then specify the location for the log file. You specify the level of detail for a log file when starting the debugger.

Service Properties

By clicking the Service Properties tab, you can view all of the service properties associated with the service template. This is useful for verifying the mapping between the input and output properties associated with a step and the service properties for a service template. The following fields are provided:

- Display Name: Display name assigned to the service property.
- Key Name: Key name assigned to the service property.
- Value: Value currently assigned to the service property.

Break Points

From the Break Points tab, you can see all of the breakpoints that are set for the current debugging session. The following details are shown:

- Flow Hierarchy: Shows the flow hierarchy indicated by a slash (/) separated path.
- Display Name: Shows the display name assigned to the step.

You can set a breakpoint to halt the flow at a particular step by selecting the step and then clicking on the breakpoint icon control.

When you no longer need the currently set breakpoints, remove them by clicking Remove All Break Points.

Managing tasks during debugging

You can control the execution and flow of tasks during the debugging process.

For more information on the options available from the Debugger view, see the following topics.

Controlling the processing flow of debug tasks

The following describes the general procedure for debugging a service template.

Before you begin

A service template in the Developing state must exist with steps you added to the Flow view.

Procedure

1. If you do not expect any problems when running the plug-ins in the service template, the first step of the debug process is to run the debug task without pausing between steps. In the **Debugger** view or the **service template debugging** view, make sure that there are no problems with the flow transitions or the plug-in processing. If the service template contains plug-ins that you do not want to run at this time, skip this step and move directly to step 2.
2. If you identify an problem with a flow transition or plug-in processing, run the steps in the debug task individually to identify the precise location and nature of the problem. You can also test the behavior of the plug-ins by assigning unexpected values to input and output properties.
   - Click **Step Into** to run the currently selected step.
   - Click **Step Over** to run the next step.
   - Click **Step Return** to run the step in the upper flow after completing the steps in the current flow.

- To start the debugging process start again from the failed step, click the debug arrow, and choose **Retry the Task Starting from the Failed Step**. By trying again from a failed step, you can resume the debug task with the same task ID and the original property values. You can use this approach when the cause of the failure is resolved. For example, you can try a step again that fails because of a temporary problem with the network when the network connection is available again.
- To start the debugging process again from the step after the failed step, click the debug arrow, and choose **Retry the Task Starting from the Step after the Failed Step**. By retrying from the step after the failed step, you can resume the debug task with the same task ID and the original property values. This approach is best in situations where there is no must to run the failed step. When you try a task again from the step after the failed step, the task processing continues as if the failed step ended normally. You can use this approach when you encounter an problem in a step, but want to continue running the debug task and handle the problem later.

Handling interruptions of debug tasks

Under specific circumstances, the tasks might be interrupted or ended in unexpected ways during a debugging session.

The steps in the debug task appear in the Flow view of the service template debugging view in the order in which they run. The icon of the step indicates the status of the step. You must be aware of how tasks are affected when the debugging process is interrupted.

Handling of debug tasks when the Ops Center Automator server stops

If the Ops Center Automator server stops during debugging, the debug task that is running is forcibly stopped. Therefore, before stopping the Ops Center Automator server, wait for all debug tasks that are still running (not yet completed) to finish or stop the debug tasks. This is the same as the handling of tasks generated when normal services are run.

Handling of debug tasks when a failover of the cluster occurs

When a cluster failover (switchover of the system) occurs during debugging, the debug task is forcibly stopped. This is the same as the handling of tasks generated when normal services are run.

Handling of debug tasks when the user logs out during debugging

When a user logs out during debugging, a confirmation dialog box appears asking for confirmation before logging out the user. If you choose to log out, the debug task is forcibly stopped.

Handling of debug tasks when the browser is closed

When the browser is closed during debugging, the debug task continues running normally. If the task is running step-by-step, the task remains at the step where it stopped. To stop the debug task, you must log on again and then stop the debug task from the Debug Task List view.

Controlling the display of tasks in the task list

While debugging tasks, you can specify how the list of tasks are managed in the Task List.

You can specify the period of time to retain tasks before they are archived and no longer shown in the Task List. After tasks are archived (and become history entries), the detailed information about the tasks is deleted, and you can no longer view it in the Tasks List view again.

You can archive tasks automatically by specifying a value in one of the following properties:

Property key	Description	Range
tasklist.autoarchive.taskRemainingPeriod	Specifies the period, expressed in days, to retain completed tasks in the task list. When the specified period has passed, the tasks are automatically archived. The automatic archiving of tasks takes place one time a day according to the time specified through the **tasklist.autoarchive.executeTime** property.	1 - 90 Default: 7
tasklist.debugger.autodelete.taskRemainingPeriod	Specifies the period, expressed in days, before debug tasks are automatically deleted.	1 - 90 Default: 7
tasklist.autoarchive.maxTasks	Specifies the maximum combined number of tasks and debug tasks to retain in the Tasks List. When the maximum number of tasks in the Tasks List is exceeded, the excess tasks are automatically archived, starting from those with the oldest end date and time. The archived tasks are managed as history entries. Debug tasks are deleted automatically and are not retained in history. Automatic archiving and automatic deletion take place one time a day at the time specified by the **tasklist.autoarchive.executeTime** property.	100 - 5000 Default: 5000

Property key	Description	Range
	When there are more tasks than the specified value, trying to run a new service results in a "maximum exceeded" error, and no task is generated. Periodic tasks that were run previously are not subject to this limit and can generate new tasks. Therefore, to allow new services to run, you must estimate the number of executions that take place per day to specify the tasklist.autoarchive.**`taskRemainingPeriod`** property.	

Verifying the property mapping of a plug-in

When debugging a service template, you can control the flow of running the tasks. The following describes the general procedure for verifying the property mapping of a plug-in.

Before you begin

A service template in the Developing state must exist with steps you added to the Flow view.

Procedure

1. In the **Flow** view, select the step whose plug-in property values you want to verify. The **Debugger** view appears the input properties and output properties of the step you selected.
2. Click the **Service Properties** tab at the bottom of the service template debugging view. The values of the service properties are displayed.
3. In the **Debugger** view, review the contents of the **Step Properties** for the plug-in property you want to verify, and identify the service property to which it is mapped.
4. In the **Key Name** column of the **Service Properties** tab, find the service property you identified in step 3.
5. In the **Debugger** view and the **Service Properties** view, make sure that the same value appears in the **Value** columns for the plug-in property and the mapped service property. If a service property is not mapped to the intended plug-in property or the values of the plug-in property and the service property differ, fix the problem in the service template editing view by clicking on the pencil icon and then supplying the correct value through the **Edit Step Property** dialog box. You can also change the values of the plug-in properties. By doing so, you can test the plug-in processing when property mapping is configured correctly to see how the processing of subsequent steps and the flow transitions change with an assortment of values.

Edit Step Property dialog box

During the debugging process, you can edit a step property if necessary.

The following table describes the **Edit Step Property** dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

Field	Subfield	Description
Name:	-	Shows the property name.
Display Name:	-	Shows the display name for the property.
Description:	-	Shows a description for the property.
Multiline:	Yes/No	Indicates whether multiline value is required for the property.
Value *	-	Shows the current value for the property.

Importing property values while debugging

While debugging a service template, you can import property values in a specified properties file.

Procedure

1. From the debugger interface, click **Import**.
2. In the displayed dialog box, enter the name of the properties file to use for storing the property values, or use the browser to search for the specified file, then click **Open**.

   When the import finishes, notifications about the property values that have and have not been imported are temporarily visible as follows:

   - Updated properties: Properties to which the values are applied.
   - Skipped properties: Properties for which the values are not applied. These are properties for which the attribute values cannot be changed and for which the values are not applied due to the property value definitions.
   - Undefined properties. Properties that are defined in the file but do not exist in the target service.

   When importing a properties file, the JSON or key=value format is supported.

   The conditions that must be met to apply property values during an import are shown in following table.

Property Group Attribute	Property Attribute				
hidden	**paramMode**	**visibility**	**reference**	**hidden**	**readOnly**
False	in	config	false	false	false
False	in	exec	false	--	--

If the properties do not meet these conditions, or there are no corresponding properties defined in the service, the values contained in the properties file are not applied. The values are also not applied if the "`value`" field is not defined or is set to null.

Note: If the length of `keyName` exceeds the limit, the property is classified as a property that does not exist in the service.

If an error occurs during an import, the error dialog box appears, and the import is canceled, leaving all property values unchanged. An error occurs when the specified file does not exist or the properties file definitions are not valid.

Properties file format

In the properties file, the property key and value used by the executed services can be defined in JSON and key=value format.

JSON format

```
{
"properties": [
      {
      "keyName": "property-key",
      "displayName" : "property-display-name"
      "description" : "description-of-property"
      "type" : "property-type"
      "value": "property-value"
      },
      {
      "keyName": "property-key",
      "displayName" : "property-display-name"
      "description" : "description-of-property"
      "type" : "property-type"
      "value": "property-value"
      },
...
]
}
```

Following are definition details for the JSON format:

- The `displayName`, `description`, and `type` fields are optional.
- When you specify `"value": ""`, the property value is set to an empty value.
- The value for the password type property can be in plain text or encrypted. The `"value"` field of the password type property is not exported for security reasons. The defined value is imported as is.
- In the properties file, define only properties for which you want to set values. The values of properties that are not defined in the imported file remain unchanged. When exporting step properties, the type field is only output for the service component.

key=value format

To specify property values for a key=value properties file, use the following format:

```
property-key=property-value [line break]
```

Following are definition details of the key=value format:

- Specify a property name and a property value on each line.
- Lines starting with a hash mark (#) are handled as comment lines.
- Lines that do not contain an equal mark "=" are handled as comment lines.
- A line break needs to be added at the end of each property setting line.
- Do not add line breaks in the middle of the property name and property value lines.
- Characters are case-sensitive.
- Even when a "\" is contained in strings like service and plug-in resource files, you do not must type "\\".
- "\" is handled as a "\".
- The characters at the beginning of a line up to the first equal sign (=) are treated as a property name.
- The characters after the equal sign (=) after the property name, until the end of the line are treated as the property value.
- The line break at the end of the properties file (EOF) is optional.
- Empty lines (lines containing line breaks only) are ignored.
- Both CR+LF and LF can be used as line breaks.
- When using the `property-key = [line break]` format, the property value is set to an empty value.

Exporting property values while debugging

While debugging, you can export property values to a properties file. This allows you to save multiple property values in a file for subsequent reference.

Procedure

1. From the debugger interface, click **Export**.
2. Access the browser to get the downloaded properties file.

   The property values are exported in the JSON format and, by default, are saved to the `service_properties.json` file.

   See Properties file format (on page 116) for more information on the format.

Releasing a service template

The final step to complete a service template is the release process. The release action changes the configuration type of a service template and the related plug-ins to Released. You can create new services and tasks from a service template in Released status.

In Released status, the service template appears in Ops Center Automator as an available service template from which you can create services.

The released service template can also be placed in the flow as a service component during the development of the service template.

Return codes of the step using a service component are as follows.

0
: The step in the service component ended normally.

1
: The step in the service component ended with a warning. There is no step that ended abnormally.

2
: The step in the service component ended abnormally.

Note: You can release a template one time only.

Before you begin

A service template in the Developing state that has completed the build process with no errors and performs as designed.

Procedure

1. From the Service Builder **Edit** window, click **Release**.
   The service template undergoes the release process and the configuration type changes to Released status.

Result

After the release process completes successfully, the service template is available in the Services tab with the Released configuration type.

- The service template is removed from the Developing state and appears under the Released tab when viewing available service templates.
- Any services created while the template was in Developing status are deleted.
- Any tasks run from the template while in Developing status are archived.
- The service template appears in the Service Template list when creating a new service.
- The related plug-ins appear in the Released tab of the Component view.
- If specified in the service definitions, a service component is created and located under the Service tab of the Component view.

Next steps

Create a new service using the released service template in the Ops Center Automator user interface.

For more information on creating services, see the *Hitachi Ops Center Automator User Guide*.

Chapter 7: Advanced options

This module covers the other functions available in managing service templates and plug-ins.

Editing the service template attributes

You can view and customize the details associated with a service template and specify the custom files that affect how it is presented and scheduled from the **Edit Service Template Attributes** dialog box.

Before you begin

Verify that the service template is in the Released or Developing state.

Procedure

1. From the Service Builder **Edit** window **General** tab, click **Edit**.
   The **Edit Service Template Attributes** dialog box opens.
2. Enter the information for the service template, then click **OK**.

Result

The changes you made are visible in the updated service template.

Next steps

Continue editing the service template by selecting plug-ins from the list, providing the input and output step properties, and establishing the flow.

Edit Service Template Attributes dialog box

When editing or creating a service template, you can view and specify additional details for the service template.

The following table describes the dialog box fields, subfields, and field groups from the Edit Service Template Attributes dialog box. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

Field	Subfield	Description
Key Name	-	The key name for the service template.
Version	-	The version of the service template.
Vendor ID	-	The vendor ID of the service template.
Display Name*	-	The name of the service template that is visible to the user.
Vendor Name	-	The vendor name of the service template.
Description	-	The description of the service template.
Tags	-	Tags associated with the service template.
Advanced Options	Icon	An image file (48 pixels × 48 pixels) using PNG format. A default image is provided but you can change the icon by clicking Change. You can return to the default icon by clicking Restore Default Icon.
	Custom File/Custom File package	The custom files give information that is visible for the **Service Details** dialog box and the overview associated with the service. Click Browse to upload a file in .html, .js, .css, or .jpeg format. If for some reason, you want to delete the custom file associated with template, click Delete.
	Service Details File Name	Specifies the name of the file that gives the information for the **Service Details** dialog box.
	Service Overview File Name	Specifies the name of the file that contains the information for the overview associated with the service template that is presented to the user.
	Available Scheduling Options:	The schedule for executing the task can be: Immediate, Schedule, or Recurrence.
	Available Action:	You can choose to allow the user to Forcibly Stop the execution of a service when the processing a step hangs for some reason or allow the user to Retry the processing of the service template from a failed step, or from the point just after the last failed step.
Fields with an asterisk (*) are required.		

Creating property groups

You can create property groups to categorize the properties associated with a service template.

You can assign the input and output properties of a service template to a custom property group. If no property group is required, the system uses the default reserved.defaultGroup. If you create a property group, the input and output properties are shown according to the custom property group name to which they are assigned.

Before you begin

A service template in Developing status must exist.

Procedure

1. From the Service Builder **Edit** window **Edit Property** tab, go to the **Add** menu and choose **Property Group**.
   The **Create Property Group** dialog box appears where you can enter the details for the new property group.
2. Enter the details for the property group and then click **OK**.
   The newly added group is visible in the property list.
3. When creating properties or variables, you can add them to the new group. You can also select and moving existing properties from the property list to the associated group.

Create Property Group dialog box

You can create a new property group for a service template from the **Create Property Group** dialog box.

The following table describes the **Create Property Group** dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

Field	Subfield	Description
ID*	-	Specifies the ID for the new property group.
Display Name*	-	Name of the new property group shown through the user interface.
Description:	-	A optional description for the new property group.
Display/Hide:	-	Specifies whether to display or hide the property group.
Custom File package:	-	Specifies a custom file for the property group.

Field	Subfield	Description

Summary Panel Rendering	-	Define a JavaScript function for display contents of summary panel.
		The following is a sample:
		`function summarize(properties, language, displayType) {`
		`  // PropertyInformation objects are stored in the properties.`
		`  var summaryContentsMap = {};`
		`  var restriction;`
		`  for (var i = 0; i < properties.length; i++) {`
		`  restriction = JSON.parse(properties[i].restriction);`
		`  if (displayType == "exec") {`
		`  if (restriction.permission != "hidden") {`
		`  summaryContentsMap[properties[i].displayName] = properties[i].value;`
		`  }`
		`  } else {`
		`  summaryContentsMap[properties[i].displayName] = properties[i].value;`
		`  }`
		`  }`
		`  return summaryContentsMap;`
		`}`

Validation Script:	-	Generates a validation file for the property group.
Fields with an asterisk (*) are required.		

Creating a validator script for verifying property group entries

If the provided validation options are not adequate for your purposes, you can create a script to perform the necessary verification. Following is an example of a validator script written in JavaScript that verifies whether a value entered by the user is a number and is less than the maximum allowable value of 2048:

```
function (properties, lang, displayType) {
    var message = [];
       var hasError = false;

    if (displayType == "exec") {
```

```
        _.each(properties, function(property) {
            if (isNaN(property.value)) {
                message.push( "value must be a number:" + property.keyName + "=" +
property.value);
                hasError = true;
                }

            if (property.value >= 2048) {
                message.push ("value must be less than 2048:" + property.keyName +
"=" + property.value);
                hasError = true;
            }
        });
    }

    if (hasError) {
        return message;
    } else {
        return
    }

}
```

The following table shows the validator script specifications for the input property:

#	Name	Description

1	Script format	function (arg1, arg2, arg3) {
		//code
		}

2	Arguments of validator	arg1: A listing of property values in Property Group. Each element is an object that has the following properties: ▪ keyName: The name of the property in string format. ▪ value: The value of the property in string format. arg2: Locale string. e.g., ja, en arg3: Operating information when script is running (Operation with task creation: exec, Editing operation of properties: config)
3	Return value of validator	Success:

#	Name	Description
		undefined or null Failure: Error message in array or string format

If the value is not a number or is larger than the specified maximum, then a message is output through the user interface.

Edit Property Group dialog box

You can edit an existing property group for a service template from the **Edit Property Group** dialog box

The following table describes the **Edit Property Group** dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

When you enter information in a dialog box, if the information is not valid, errors that include a description of the problem appear at the right side of the box.

Field	Subfield	Description
ID:	-	Specifies the ID for the property group.
Display Name:	-	Name of the property group visible through the UI.
Description:	-	A optional description for the property group.
Display/Hide:	-	Specifies whether the property group is visible or hidden.
Custom Files package:	-	Specifies a custom files for the property group.
Summary Panel Rendering Script:	-	Defines a JavaScript function for display contents of summary panel.
Validation Script:	-	Specifies a validation file for the property group.
Fields with an asterisk (*) are required.		

Managing versions

Service Builder applies the same method of managing versions to both service templates and plug-ins. The system assigns a version number when you create a new service template or plug-in.

All service templates and plug-ins need a version number in nn.nn.nn format (*major-version-number*.*minor-version-number*.*revision-number*). Specify each of the numbers in the range from 00 through 99.

When you copy a service template or plug-in with the same Key Name and Vendor ID, you must assign the copy a new version number. You cannot duplicate a service template or plug-in with the same Plug-in key name, Vendor ID, and Version Number.

If two or more service templates with same service template key name and Vendor ID exist, only the latest version of service template is visible. Similarly, if two or more plug-ins with same Plug-in key name and Vendor ID exist, only the latest version is visible.

Under some conditions, when plug-ins are updated, a service template can include older versions that you must update to the most recent versions. In this case, you can update individual plug-ins or all the plug-ins for a service template by following these steps:

1. From the Service Builder **Edit** window, go to the Actions menu and select Component Version Management. The **Component Version Management** dialog box appears.
2. To update all of the plug-ins associated with the current service template to the latest versions, select the All Apply tab. You can see all of the plug-in component steps that will be upgraded by clicking Step list to be applied. If you prefer to individually specify each of the plug-in components that to update, select the Individual Apply tab and then select the version to apply for a step.
3. After selecting the components to update, click Apply and a message confirms that the version was updated.
4. When you finish updating versions, click Close.
5. After updating a plug-in version, save the plug-in to apply the changes you made.

Note: Before updating the service, ensure that the service template and all associated tasks are in the Released state and that the service is archived or deleted.

Component Version Management dialog box

You can manage versions of the component steps associated with a service template to ensure that you are using the most current version of a component. You have a choice of updating all of the components, or you can select specific components.

The following table describes the **Component Version Management** dialog box fields, subfields, and field groups. A *field group* is a collection of fields that are related to a specific action or configuration.

Field	Subfield	Description
Apply to All	-	Collectively updates all step components to the latest version.

Field	Subfield	Description
Individual apply	-	Selectively specifies specific step components that are updated to the latest version. When choosing this option, a list of components is shown along with details associated with the component and a step list where you can choose the components to be updated from the Versions menu. To see more details for the component, click View to open the Service Builder View window or View Plug-in Details dialog, and click the Property tab to see all properties associated with the component.
Step list to be applied	-	Shows the step list that includes the Step Name, Current Component Name, Latest Version, Latest Component Name, and Status. When clicking Latest Component Name, the Component Preview dialog is displayed and more details for the specified component are shown. You can also click View in the Component Preview dialog to open the Service Builder View window or View Plug-in Details dialog, and click the Property tab to see all properties associated with the component.

Appendix A: Reference information

This module describes the built-in service templates and plug-ins, reserved properties, and locale settings for plug-ins.

List of built-in service templates

A collection of preconfigured service templates is provided with Service Builder.

The following service templates are provided by default and are available for submitting services with no additional configuration:

Add Host to Cluster in vCenter
Allocates existing volumes used as datastores by ESX cluster hosts to a new ESX host.

Allocate Fabric Aware Volumes and Create Datastore for ESX Cluster
Allocates volumes to VMware ESX cluster hosts, configures zoning, and creates a VMware datastore under a datastore cluster.

Allocate Fabric Aware Volumes with Configuration Manager
Allocates sets of volumes from the associated infrastructure group through Ops Center API Configuration Manager for use by servers running a generic application. This service accesses the switch management server and acquires existing fabric configuration and zoning information when allocating new volumes to the host.

Allocate Volumes, Fabric, and Datastore for ESXi Host
Intelligently allocates volumes to the VMware vSphere server (ESXi host), configures zoning, and creates VMware datastores using Ops Center API Configuration Manager.

Allocate Volumes with Clone/Snapshot
Allocates sets of volumes with in-system replication (Thin Image, ShadowImage) from the associated infrastructure group through Ops Center API Configuration Manager to be consumed by servers running a generic application.

Allocate Volumes with 2DC Remote Replication
Intelligently allocates by using sets of volumes from the associated infrastructure group for use by servers running a generic application and creates a new copy topology for remote replication.

Allocate Volumes with Remote Replication (Global-Active Device)
Allocates sets of volumes with in-system replication (global-active device) from the associated infrastructure group through Ops Center API Configuration Manager to be consumed by servers running a generic application.

Allocate Volumes with Smart Provisioning
Intelligently allocates by using sets of volumes from the associated infrastructure group through Ops Center API Configuration Manager for use by servers running a generic application.

Call ServiceNow Table API
Calls the ServiceNow Table API.

Clean up Online Migration Pair
Enables you to clean up the resources created by the Create Online Migration Pair task.

Create Online Migration Pair
Runs from the creation of zones to the creation of copy pairs for online host migration through Configuration Manager. After this service is complete, you must submit the Migrate Data for Online Migration Pair service to complete the migration.

Create Online Migration Pairs for Multiple Hosts
Automatically submits a Create Online Migration Pair service task for each specified host. After the auto-submitted tasks are complete, you must submit the Migrate Data for Online Migration Pair service for each task to complete the migration.

Create ServiceNow Incident Ticket
Creates a ServiceNow incident ticket.

Global-Active Device Setup
Creates virtual storage machines, assigns Quorum Disk IDs, creates remote paths, and allocates command devices to create global-active devices.

Migrate Data for Online Migration Pair
Runs from the swap of copy pairs to the deletion of source volumes for online host migration through Configuration Manager. Before submitting this service, the Create Online Migration Pair service must be completed.

Remove Host from Cluster in vCenter
Unmounts VMFS datastores, unallocates volumes from the specified ESX host, and deletes zoning.

Retrieve ServiceNow Incident Tickets
Retrieves ServiceNow incident tickets.

Update ServiceNow Incident Ticket
Updates a ServiceNow incident ticket.

Note: The File type properties for these service templates include connection details for various components and should not be changed as the templates might no longer function properly.

Warning: Some properties associated with the built-in service templates include internal data as indicated by "Do not change." Do not change these properties.

List of built-in plug-ins

A collection of plug-ins are provided with Ops Center Automator that you can use to create customized service templates.

Note: Use only the plug-ins that are listed in this section.

The following plug-ins are included by default:

Abnormal-End Plug-in
Manages abnormal termination of flows, tasks, hierarchical flows, and repeatedly run flows.

Branch by Property Value Plug-in
Creates a conditional branch that compares and determines service property values to branch the flow of processing.

Branch by ReturnCode Plug-in
Creates a conditional branch that compares and determines the return value of the previous step to branch the flow of processing.

Email Notification Plug-in
Connects to an SMTP server and sends emails to the specified destination.

File Export Plug-in
Exports the input content to the specified output file in a format you can define by using VTL (Velocify Template Language).

File-Transfer Plug-in
Sends a file or a folder to a remote host, or receives a file or a folder from a remote host. SCP or SFTP is used as the transfer protocol.

Flow Plug-in
Creates a flow that has a hierarchy defined by nested flows.

General Command Plug-in
Runs a command line on a remote destination host.

Interval Plug-in
Controls the interval between steps by specifying the wait time.

JavaScript Plug-in
Runs any script written in JavaScript.

Python Plug-in
Runs Python scripts on the Ops Center Automator host.

Repeated Execution Plug-in
Runs a specific flow repeatedly. You can use this plug-in to implement loop processing.

Standard Output Plug-in
Outputs a specified value to the standard output. Do not use this plug-in for a new service template because this plug-in is a compatibility plug-in with the service template that was created with the former procedure.

Terminal Command Plug-in
Runs a command line on the remote destination host that is connected with Telnet or SSH using the Terminal Connect Plug-in.

Terminal Connect Plug-in
Enables terminal connections with the remote destination hosts using Telnet or SSH authentication.

Terminal Disconnect Plug-in
Disconnects the terminal from the remote destination host, which was connected with Telnet or SSH using the Terminal Connect Plug-in.

Test Value Plug-in
Compares service property values and returns 0 if the values match the specified conditions.

User-Response Wait Plug-in
Suspends processing and waits for a user response. Using this plug-in, the operator manually selects whether to continue processing.

Web Client Plug-in
Sends HTTP request messages such as GET/POST and receives HTTP response messages. When requested, it accesses the server through a proxy to complete server and proxy authentication. For example, you can use this plug-in when connecting to the Web using the REST API.

List of reserved properties

A reserved property is a special service property whose property key has a specific definition or purpose in Ops Center Automator.

The property key of a reserved property begins with reserved. You can use reserved properties by mapping them to plug-in properties in the **Specify Component Input Property Mapping Parameters** dialog box or the **Specify Component Output Property Mapping Parameters** dialog box. Users cannot define or assign values to reserved properties.

When you map a reserved property to an input property, the value of the reserved property is assigned to a plug-in property when the plug-in is run. Alternatively, select the Direct Input option, and in the Value field, specify the reserved property in the format ?dna_reserved-property-key?. In this case, the value of the reserved property supplies part of the value of the plug-in properties at plug-in execution.

When you use a reserved property as an output property, the reserved property stores the value of a designated plug-in property. By selecting the View Property option in the **Specify Component Output Property Mapping Parameters** dialog box, you can specify a reserved property to which the value of the output property is passed.

Following is a list of reserved properties:

Reserved Property Key	Description
reserved.external.hcmds.dir	Indicates the folder of Hitachi Command Suite. Example: `C:\Program Files\HiCommand`
reserved.external.path	Indicates the path of the REST API to access the dynamic data. For example, in the case of "http://localhost:22015/Automation/v1/objects/ExternalResources/IPAddresses?host=myHost&type=vm", the value becomes "/IPAddresses". The query parameter is not included.
reserved.external.query	Indicates the query parameter of the path passing to REST API. For example, in the case of "http://localhost:22015/Automation/v1/objects/ExternalResources/IPAddresses?host=myHost&type=vm", the value becomes "host=myHost&type=vm".
reserved.external.resource.dir	Indicates the resource folder of the external resource of the dynamic data.
reserved.external.userName	The user name who is logged in to Ops Center Automator from REST API.
reserved.loop.index reserved.loop.indexN	References a numerical value from 1 to 99 that indicates how many times a Repeated Execution Plug-in, that is one (or N) levels above the base step, has repeated.
reserved.loop.input reserved.loop.inputN	References the value of the inputProperties input property of a repeated execution plug-in that is one (or N) levels above the base step. Of the comma-delimited values specified in the input properties of the repeated execution plug-in, this property stores the value of the element that corresponds to the current iteration of the flow. For example, if the input property is A,B,C, the values A, B, and C are input in the order corresponding to the repetition count of the flow. The repeated execution plug-in can be run a maximum of 99 times.

Reserved Property Key	Description
reserved.loop.output	Passes values to the outputProperties output property of a repeated execution plug-in. The values output to this property are assigned to the output property as a comma-separated value. For example, if the values of the output property of the plug-in are X, Y, and Z for successive iterations, the value X,Y,Z is assigned to the output property.
reserved.service.name	References the name of the service from which a task was generated. To reference this reserved property, specify the property key in the format ? dna_reserved.service.name?. You can use this property in any plug-in to which service properties can be mapped.
reserved.service.serviceGroupName	References the service group in which the service from which a task was generated is registered. To reference this reserved property, specify the property key in the format ? dna_reserved.service.serviceGroupName?. You can use this property in any plug-in to which service properties can be mapped.
reserved.step.path	References the ID of the step that is currently being run. To reference this reserved property, specify the property key in the format ?dna_reserved.step.path?. The value of this property is the same as the step ID visible in the messages output to the task log when plug-in execution begins and ends. You can use this property in any plug-in to which service properties can be mapped.
reserved.step.prevReturnCode	Supplies the return value of the preceding step (the step that is the origin of the relational line connected to the plug-in). To reference this reserved property, specify the property key in the format ? dna_reserved.step.prevReturnCode?. If there are multiple preceding steps, the property is assigned the logical sum of all the return values. If there is no preceding step, 0 is assigned. You can use this property in any plug-in to which service properties can be mapped. If you try a task again from a step that references this reserved property without executing the preceding step, the return value from the last time the preceding step was run is set in this reserved property as the return value of the preceding step.

Reserved Property Key	Description
reserved.task.description	Supplies the description of a task. To reference this reserved property, specify the property key in the format ?dna_reserved.task.description?. You can use this property in any plug-in to which service properties can be mapped.
reserved.task.dir	Supplies the path of the temporary data folder created during task execution. This property shows a unique folder path at execution of each task. The folder referenced by this property is created on the Ops Center Automator server when the task is run, and deleted when the task is archived. Note that files and folders that start with task are reserved in Ops Center Automator, and cannot be created by the user.
reserved.task.id	Supplies the task ID. To reference this reserved property, specify the property key in the format ? dna_reserved.task.id?. You can use this property in any plug-in to which service properties can be mapped.
reserved.task.name	Supplies the task name. To reference this reserved property, specify the property key in the format ? dna_reserved.task.name?. You can use this property in any plug-in to which service properties can be mapped.
reserved.task.submitter	Supplies the user ID of the user who submitted the task for execution. If the task was retried, this property references the user ID of the user who submitted the task, not the user who retried the task. To reference this reserved property, specify the property key in the format ?dna_reserved.task.submitter?. You can use this property in any plug-in to which service properties can be mapped.
reserved.task.tags	A reserved property to reference the tags set for a task.
reserved.task.url	Supplies the URL for accessing the **Task Details** dialog box. To reference this reserved property, specify the property key in the format ?dna_reserved.task.url?. You can use this property in any plug-in to which service properties can be mapped.

Reserved Property Key	Description
reserved.terminal.account	References the user ID used by a terminal connect plug-in. This property is used by the commandLine input property of a terminal command plug-in. It stores the login name of the user account used to connect to the terminal.
reserved.terminal.password	References the password used by a terminal connect plug-in. This property is used by the commandLine input property of a terminal command plug-in. It stores the password of the user account used to connect to the terminal.
reserved.terminal.suPassword	References the root password used by the terminal connect plug-in. This property is used by the commandLine input property of a terminal command plug-in. It stores the password of the superuser used to connect to the terminal.

Locale settings for plug-ins

The locale setting that applies to a device on which an action is run by a plug-in depends on the OS. The following describes the locale settings applied when plug-ins are run in each OS.

In Windows

When a script or command is run on a target device, the locale and character set of the target device should match those of Ops Center Automator. The locale and character set are determined by the settings in the Region and Language area of the Windows Control Panel that govern date and time formats, user-level display languages, system-level display languages, and system locale settings.

In Linux OS

The locale setting applied while the plug-in runs depends on the Character Set Auto Judgment setting in the Create Custom Plug-in dialog box or the Edit Custom Plug-in dialog box.

- If the check box is disabled, scripts are run with the `LC_ALL=C` locale. Make sure that commands and command parameters consist only of ASCII characters. If a command parameter, standard output, or standard error output contains non-ASCII characters, the characters might become garbled and prevent the command from running normally.
- If the check box is enabled, the script is run using the default locale of the connecting user. When running a script or command on an operation target device, the environment variable `LC_ALL` and `LANG` are set to the default locale of the connecting user. No other environment variables are changed.

When running a script or a command the locale is assigned in the following order of priority:

- 1: Value of `LC_ALL`
- 2: Value of `LC_CTYPE`
- 3: Value of `LANG`

Appendix B: Description of built-in plug-ins

This module describes the plug-ins that are preconfigured in Automator Service Builder.

General Command Plug-in

The General Command Plug-in runs a command line on the destination host.

If you have pre-set authentication information in the Agentless Remote Connections view, you can run commands by specifying the following information in the general command plug-in:

- Device on which to run the command (destinationHost property)
- Command to run (commandLine property)
- Command arguments (commandLineParameter property)

If destination host is a Ops Center Automator Server (localhost), the user ID and the password are unnecessary.

For the command to run on the target device, specify characters that can be used in commands in the OSs of the Ops Center Automator server and the operation target device. For example, if the Ops Center Automator server and the target device both run the Windows OS with Japanese Locale, you can specify characters in the MS932 character set.

When local execution function is enabled and the OS of the local host is Windows, the command runs with System account privileges. If the OS is Linux, the command runs with root user privileges. The execution folder for the command is specified as follows:

- When the connection destination is running Windows: `Admin$\Hitachi\CMALib\HAD\home Admin$` is the folder specified in the windir environment variable.
- When the connection destination is running Linux OS and true is specified for the elevatePrivileges property: The home folder of the root user
- When the connection destination is running Linux OS and false is specified for the elevatePrivileges property: The home folder of the connection user

Prerequisites

Specific commands must be installed on the OS of the target device before you use the general command plug-in.

To use the general command plug-in when the target device is running Windows, you must enable administrative sharing.

Cautionary notes

- IPv6 is not supported when Ops Center Automator is running on Linux OS and the destination host is Windows.
- The locale and character set at the for running the plug-in depend on the OS of the operation target device.
- If a task is stopped while the plug-in is running, the status of the task changes to Failed or Completed when the general command plug-in processing finishes. The status of steps and tasks after plug-in finishes running depends on the return code of the step and the condition for running subsequent steps. You can set a Subsequent-step Execution Condition in the **Create Step** or **Edit Step** dialog box.
- The execution method differs depending on the OS of the target device. The command is run by SMB and RPC in Windows, and SSH in Linux OS. Therefore, the SSH server must be set up on Linux-based operation target devices.
- The SSH port number is set in the connection-destination properties file (`connection-destinationname.properties`) or the properties file (`config_user.properties`).
- When the target device is running Windows, user profiles are not inherited. This means a plug-in can produce different results from a command or script run on the desktop. To avoid this problem, do not reference settings in user profiles, such as user environment variables and registry entries, and proxy settings, when running a plug-in. If a command or script references an element of a user profile, the command or script might not behave as expected. For example, when you run a command or script that references proxy settings, the command or script might fail with a communication error. This might occur in scenarios such as implementing a Windows Update using a script.
- If the target device is running Linux OS, and you must specify non-ASCII characters in the commandLine or commandLineParameter property, the login script setting is required in the following plug-ins:
  - General Command Plug-in
  - File-Transfer Plug-in
  - Custom Plug-in
  - Terminal Command Plug-in

Note: Applies only to the File-Transfer Plug-in when a non-ASCII character is included in the value of the remoteFilePath property.

Login script setting

You can check the status of the `istrip` setting in standard output by running the **`stty -a`** command in the command line for each plug-in. If `-istrip` appears in standard output, the `istrip` setting is disabled. If `istrip` is not prefixed with "**-**" in standard output, the `istrip` setting is enabled.

The following example shows "istrip" with the correct setting:

```
[root@vm03 /]# stty -a
speed 38400 baud; rows 19; columns 80; line = 0;
intr = ^C; quit = ^¥; erase = ^H; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd -cmspar cs8 -hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany -imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
```

If `istrip` appears without the dash (-) you must edit the login script file and add the line `stty -istrip`.

Note: If you use the setting that elevates user permissions to root, the `istrip` setting will be overwritten in the login script of the root user. Make sure that the `istrip` setting is disabled in the login script of the root user.

Return codes

The General Command Plug-in generates the following return codes:

Return Code	Description
0-63	The return code (0 to 63) of the command or script is returned as the return code of the plug-in. The meaning of the command or script depends on the command or script.
64	If the return code of the command or script is 64 or higher, 64 is returned as the return code of the plug-in.
65	The connection with the Ops Center Automator server failed. For example, the Ops Center Automator server might have stopped while the plug-in was running.
66	The following user is mapped to the Ops Center Automator user: ▪ A user who does not belong to the Administrators group. ▪ A user other than the built-in Administrator who belongs to the Administrators group, in an environment with UAC enabled.
68	No information about the target job execution ID exists.
69	An environment variable of the task-processing engine cannot be acquired.
70	The connection with the operation target device failed.
71	Command execution failed.
72	The execution status of the command cannot be acquired.

Return Code	Description
76	The connection timed out.
77	The host name of the operation target device cannot be resolved.
78	Authentication with the operation target device failed for one of the following reasons: ▪ Password authentication failed. ▪ Public key authentication is not set up on the operation target device. ▪ During authentication of the public key, the private key did not match the pass phrase. ▪ During authentication of the public key, the private key did not correspond to the public key registered in the operation target device. ▪ During authentication of the public key, a private key was used that is not valid.
80	Task execution has stopped.
81	The plug-in was called in a status that is not valid.
82	The request message from the task-processing engine cannot be correctly parsed.
83	The Ops Center Automator server environment is corrupted.
84	Information about the plug-in cannot be obtained.
86	The specified property value is not valid.
127	Another error occurred.

Property list

The following properties are available for the General Command plug-in:

Property key	Property name	Description	I/O type
destinationHost R	Destination Host	Specify the IPv4 address, IPv6 address, or host name of the operation target device. The host name cannot exceed 1,024 characters. The Ops Center Automator server and the operation target device must be connected by a network. Note that multiple IP addresses or host names cannot be specified.	Input

Property key	Property name	Description	I/O type
credentialType [R]	Credentials Type	As the authentication type to use during command or script execution, specify either of the following: **Destination** Specify this option to use the authentication information set in the Agentless Remote Connections view. Specifying destination applies the authentication information set for Windows or SSH in the connection destination definition according to the IP address of the Ops Center Automator logon user. You can omit the specification of properties relating to authentication information (account, password, suPassword, publicKeyAuthentication), and keyboardInteractiveAuthentication. **Property** Specify this option to use the values specified in the following properties as authentication information: ▪ account ▪ password ▪ suPassword ▪ publicKeyAuthentication ▪ keyboardInteractiveAuthentication	Input
account	User ID	Specify the user ID to use to log on to the operation target device, using a maximum of 256 characters.	Input

Property key	Property name	Description	I/O type
		You can also specify a domain user in either of the following formats: ▪ domain-name \ user-name ▪ user-name @ domain-name	
password	Password	Specify the password to use to log on to the operation target device, using a maximum of 256 characters. You can omit this property when the operation target device is running Linux OS and true is specified for the publicKeyAuthentication property.	Input
suPassword	Root Password	If the OS of the operation target device is Linux OS, specify the root password using a maximum of 256 characters. If the OS is Windows, this property does not need to be specified. You can also omit this property when the operation target device is running Windows, or when false is specified for the elevatePrivileges property.	Input
runAsSystem	Run As System Account	When running on a Windows host, you can specify true to run the command using the system account or false to allow the command to run using another connected user. The connected user becomes the user specified for Agentless Remote Connections or plug-in properties, regardless of the specification of this property. The values are not case sensitive. If you do not specify a value, false is assumed. If the OS of the destination host is not Windows, this value is ignored. The default value is false.	Input
publicKeyAuthentication	SSH public key authentication settings	If the OS of the operation target device is Linux OS, specify either of the following depending on	Input

Property key	Property name	Description	I/O type
		whether you want to use public key authentication. The values are not case sensitive. If you do not specify a value, false is assumed. You can omit this property when the operation target device is running Windows. **true** Specify this option to use public key authentication. **false** Specify this option to use password or keyboard interactive authentication. The default value is false.	
keyboardInteractiveAuthentication	SSH keyboard interactive authentication settings	Controls whether to use SSH keyboard interactive authentication for the Linux OS environment. If the OS of the destination is Linux OS, the system toggles between using and not using keyboard interactive authentication. If the property is set to true, keyboard interactive authentication is used. If the property is set to false, keyboard interactive authentication is not used. This property is not case-sensitive. This property is valid only when publicKeyAuthentication is set to false. If this property does not exist (which is true for a previous plug-in version) or if no value is specified, false is assumed for the property. The default value is false.	Input
elevatePrivileges	Elevate Privileges	If the OS of the operation target device is Linux, specify either of the following depending on whether you want to elevate the user to root privileges. The values are not case sensitive. If you do not specify a value, true is assumed. You can omit this	Input

Property key	Property name	Description	I/O type
		property when the operation target device is running Windows. **true** Specify this option to run commands as a user with root privileges. **false** Specify this option to run commands without elevating the user to root. Commands are run with the privileges of the connection user. The default value is false.	
commandLine R	Command Line	Specify the absolute path of the command or script to be run on the operation target device, using a maximum of 256 characters. In Windows, the execution user is changed according to the specification of the plug-in property "runAsSystem". Special characters that represent environment variables in the command line are not escaped. To handle a special character as a character string, escape the character with a percent sign (%) in Windows, and a backslash (\) in Linux OS. The command or script is run subject to the privileges of the following user:	Input

Property key	Property name	Description	I/O type
		When the operation target device is running Windows ▪ If destination is specified for the credentialType property, the command is run with the privileges of the user set in the Agentless Remote Connections view. ▪ If property is specified for the credentialType property, the command is run with the privileges of the user specified in the account property. When the operation target device is running Linux OS ▪ If destination is specified for the credentialType property, the command is run with the privileges of the root user or the user set in the Agentless Remote Connections view, depending on the value of the elevatePrivileges property. ▪ If property is specified for the credentialType property, the command is run with the privileges of the root user or the user specified in the account property, depending on the value of the elevatePrivileges property.	
commandLineParameter	Command-line Parameters	Specify the arguments of the command or script using a maximum of 1,024 characters.	Input

Property key	Property name	Description	I/O type
		As the command line parameters, specify characters that can be entered in command lines in the OSs of the Ops Center Automator server and the operation target device. Special characters that represent environment variables in the command line are not scaped. To handle a special character as a character string, escape the character with a percent sign (%) in Windows, and a backslash (\) in Linux OS. You can also specify an environment variable as the value of a command line parameter. The specification format depends on the OS of the operation target device. If the operation target device is running Windows: % environment-variable % If the operation target device is running Linux OS: $ environment-variable	
outputCondition	Output Condition	Specifies a condition to be output to the standard output property 1-3. You can specify the following values: ▪ always -- Outputs a null character even if it does not match the specified pattern. ▪ patternMatch -- Outputs only when matching the standard output pattern 1-3. If there is no output in output properties, mapped service properties are also not updated. The default value is Always.	Input
stdoutProperty1	Standard Output Property 1	The character string extracted by the stdoutPattern1 property is output to this property.	Output

Property key	Property name	Description	I/O type
stdoutPattern1	Standard Output Pattern 1	Specify the regular expression pattern of the standard output to output to the stdoutProperty1 property, using a maximum of 1,024 characters. Specify the regular expression pattern in a PCRE-compliant format. If you specify the key of a service property in the stdoutProperty1 property but do not specify the stdoutPattern1 property, the entire standard output and standard error output of the command or script specified in the commandLine property is assigned to the service property.	Input
stdoutProperty2	Standard Output Property 2	The character string extracted by the stdoutPattern2 property is output to this property.	Output
stdoutPattern2	Standard Output Pattern 2	Specify the regular expression pattern of the standard output to output to the stdoutProperty2 property, using a maximum of 1,024 characters. Specify the regular expression pattern in a PCRE-compliant format. If you specify the key of a service property in the stdoutProperty2 property but do not specify the stdoutPattern2 property, the entire standard output and standard error output of the command or script specified in the commandLine property is assigned to the service property.	Input
stdoutProperty3	Standard Output Property 3	The character string extracted by the stdoutPattern3 property is output to this property.	Output

Property key	Property name	Description	I/O type
stdoutPattern3	Standard Output Pattern 3	Specify the regular expression pattern of the standard output to output to the stdoutProperty3 property, using a maximum of 1,024 characters. Specify the regular expression pattern in a PCRE-compliant format. If you specify the key of a service property in the stdoutProperty3 property but do not specify the stdoutPattern3 property, the entire standard output and standard error output of the command or script specified in the commandLine property is assigned to the service property.	Input
[R]: Required			

The standard output or standard error output of the commands or scripts specified in these properties are output as the standard output of the step in Ops Center Automator. However, processing for which the total standard output and standard error output of the command or script exceeds 100 KB is outside the scope of product support. Run the command or script in advance to make sure that the total standard output and standard error output does not exceed 100 KB.

If the operation target device is running Windows, the content specified in the commandLine and commandLineParameter properties are made into a batch file and run on the operation target device. Therefore, the result of this action might differ from the result if the same command and script were run from the command prompt.

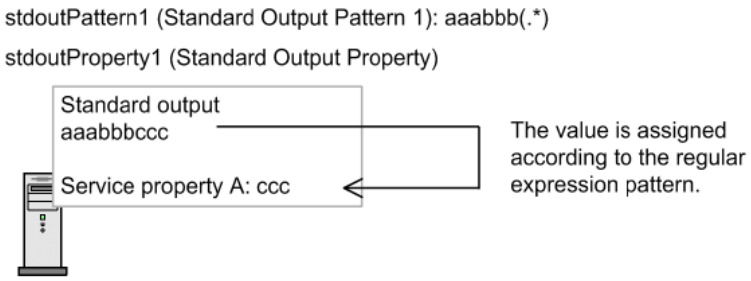
If the operation target device is running Linux OS, linefeed codes in standard output and standard error output are changed as follows:

- CR(0x0d) is changed to LF(0x0a)
- CR+LF(0x0d0a) is changed to LF+LF(0x0a0a)

In addition, if the character string at the end of the standard output and standard error output is not a linefeed code (CR, LF, or CR+LF), LF(0x0a) is added to the end.

Usage example of stdoutPattern and stdoutProperty properties

By using the stdoutPattern property, you can extract the value output to standard output and store it in the stdoutProperty property. The following figure shows the data flow when specifying aaabbb(.*) in stdoutPattern1.



As defined in stdoutPattern1, for the standard output aaabbbccc, the value after aaabbb (in this case ccc) is extracted. The extracted value is stored in the stdoutProperty1 property.

Specifying the SSH port number

You can specify a port number when using SSH to connect to the operation target device. The following table describes how to specify the port number and priority of each method.

Priority	Set in	Property key	Default value
1	Connection-destination properties file (`connection-destination-name.properties`)	ssh.port	--
2	Properties file (`config_user.properties`)	ssh.port.number	22

File-Transfer Plug-in

The File-Transfer Plug-in transfers a file or folder to or from a remote host.

If you have pre-set authentication information in the Agentless Remote Connections view, you can run the File-Transfer Plug-in by specifying the following information:

- operation target device (remoteHost property)
- Transfer mode (transferMode property)
- Path of a file or folder on the Ops Center Automator server (localFilePath property)
- Path of a file or folder on the operation target device (remoteFilePath property)

In the file path for forwarding to the agentless connection destination, specify characters that can be used in commands in the OSs of the Ops Center Automator server and the operation target device. For example, if the Ops Center Automator server and the operation target device are both running the Windows OS with Japanese Locale, characters in the MS932 character set can be specified.

If destination host is a Ops Center Automator server (`localhost`), the user ID and the password are unnecessary.

If the operation target device is running Windows, the file is transferred by the user set in the authentication information.

If the operation target device is running Linux OS, the file is transferred subject to the privileges of the root user or the connection user, depending on the value of the elevatePrivileges property.

Note: If the local execution function is enabled, the file is not forwarded. If the OS of the local host is Windows, the file is copied to the local host with the privileges of the System account. If the OS of the local host is Linux, the file is copied to the local host with root user privileges.

Prerequisites

Depending on the OS of the operation target device, configure the environment as follows:

For Windows

- Make sure that the Ops Center Automator server and operation target device can communicate using the specified ports.
- Before running the file-forwarding plug-in, enable administrative sharing on the operation target device.

For Linux OS

- You can set the SSH port number in the connection-destination properties file (`connection-destination-name.properties`) or the properties file (`config_user.properties`).
- On the operation target device, install an SSH server that supports SCP or SFTP.

Specific commands must be installed in the OS of the operation target device before you use the File-Transfer Plug-in.

Cautionary notes

- IPv6 is not supported when Ops Center Automator is running on Linux OS and the destination host is Windows.
- The execution method differs depending on the OS of the operation target device. File transfer is implemented by RPC and CIFS (SMB) in Windows, and SSH (SCP or SFTP) in Linux OS. When selecting a protocol in the definition of an agentless connection destination, select Windows in Windows and SSH in Linux OS. Note that the feature to suspend and resume file transfers using SFTP is not supported.
- The maximum total size of all transferred files is 4 GB.
- The maximum number of files and folders that can be transferred at a time is 10,000.
- If a received file has the same name as a file that exists locally, the system might try to overwrite the file. However, if the file to be overwritten has the attribute Read only, Hidden file, or System file, the file cannot be overwritten and file transfer fails.
- You cannot specify a Windows UNC path or a network drive as the source or destination of a file transfer.

- On the machine where Ops Center Automator is installed and the connection-destination host, in addition to the free space needed for the files and folders themselves, an amount of free space equivalent to twice the size of the transferred files is required as a temporary work area. The temporary work area is as follows:
  - For the machine where Ops Center Automator is installed (non-cluster environment): The drive where Ops Center Automator is installed.
  - For the machine where Ops Center Automator is installed (cluster environment): The shared disk.
  - When the connection-destination is running Windows: The system drive.
  - When the connection-destination is running Linux OS: The folder specified in the plugin.remoteCommand.workDirectory.ssh key in the properties file config_user.properties).
- The limitations of the OS override those set in the Ops Center Automator system. Examples of these limitations include the maximum size of a file, the number of files per folder, the length of file and folder names, and the resources available to the user. File forwarding that exceeds the limitations of the OS is outside the scope of product support. The OSs whose limitations affect Ops Center Automator are those on the Ops Center Automator server and on operation target devices. The OS limitations that govern which resources are available to users are those set for the connection user and for users with root privileges. Limitations for users with root privileges only apply in Linux OS.
- When you specify a folder on a host running Linux OS as the file-transferring destination, the process might fail if the total size of the files in the folder exceeds the maximum permitted size for one file. The maximum size for one file is governed by file system restrictions and OS limitations that apply to the resources available to the user. Ops Center Automator archives files before sending them, which means that the limits of the destination host might be exceeded despite the individual files in the archive being smaller than the maximum size. In this scenario, either reduce the total size of the files in the folder you are sending, or increase the limits at the destination.
- If execution of a task is stopped during plug-in execution, the status of the task becomes Failed or Completed when the processing of the File-Transfer Plug-in has finished. The status of steps and tasks after plug-in execution has finished depends on the return code of the step and the condition for executing subsequent steps. You can set Subsequent-step Execution Condition in the Create Step dialog box or the Edit Step dialog box.

Return codes

The following return codes are generated by the File-Transfer Plug-in:

Return Code	Description
0	Ended normally.
65	The connection with the Ops Center Automator server failed. For example, the Ops Center Automator server might have stopped while the plug-in was executing.

Return Code	Description
66	The following user is mapped to the Ops Center Automator user: ▪ A user who does not belong to the Administrators group. ▪ A user other than the built-in Administrator who belongs to the Administrators group, in an environment with UAC enabled.
68	No information about the target job execution ID exists.
69	An environment variable of the task-processing engine cannot be acquired.
70	The connection with the operation target device failed.
71	An attempt to call a command on the operation target device failed.
72	The execution status of the command cannot be acquired.
73	The file or folder cannot be transferred.
76	The connection timed out.
77	The host name of the operation target device cannot be resolved.
78	Authentication with the operation target device failed for one of the following reasons: ▪ Password authentication failed. ▪ Public key authentication has not been set up on the operation target device. ▪ When the public key was being authenticated, the private key did not match the pass phrase. ▪ When the public key was being authenticated, the private key did not correspond to the public key registered in the operation target device. ▪ When the public key was being authenticated, a private key was used that is not valid.
80	Task execution has stopped.
81	The plug-in was called in a status that is not valid.
82	The request message from the task-processing engine cannot be correctly parsed.
83	The environment of the Ops Center Automator server is corrupted.
84	Information about the specified plug-in cannot be obtained.
86	The specified property value is not valid.
127	Another error has occurred.

Property list

The following properties are available for the File-Transfer Plug-in:

Property key	Property name	Description	I/O type
remoteHost R	Remote Host	Specify the IPv4 address, IPv6 address, or host name of the operation target device. The host name must be within 1,024 characters. The Ops Center Automator server and the operation target device must be connected by a network. Note that multiple IP addresses or host names cannot be specified.	Input
credentialType R	Credentials Type	As the authentication type to use during command or script execution, specify either of the following: **Destination** Specify this option to use the authentication information set in the Agentless Remote Connections view. Specifying destination applies the authentication information set for Windows or SSH in the connection destination definition according to the IP address of the Ops Center Automator logon user. You can omit the specification of properties relating to authentication information (account, password, suPassword, publicKeyAuthentication), and keyboardInteractiveAuthentication. **Property** Specify this option to use the values specified in the	Input

Property key	Property name	Description	I/O type
		following properties as authentication information: ▪ account ▪ password ▪ suPassword ▪ publicKeyAuthentication ▪ keyboardInteractiveAuthentication The default value is destination	
account	User ID	Specify the user ID to use to log on to the operation target device, using a maximum of 256 characters. You can also specify a domain user in either of the following formats: ▪ domain-name \ user-name ▪ user-name @ domain-name	Input
password	Password	Specify the password to use to log on to the operation target device, using a maximum of 256 characters. You can omit this property when the operation target device is running Linux OS and true is specified for the publicKeyAuthentication property.	Input
suPassword	Root Password	If the OS of the operation target device is Linux, specify the root password using a maximum of 256 characters. If the OS is Windows, this property does not need to be specified. You can also omit this property when the operation target device is running Windows, or when false is specified for the elevatePrivileges property.	Input

Property key	Property name	Description	I/O type
publicKeyAuthentication	SSH public key authentication settings	If the OS of the operation target device is Linux, specify either of the following depending on whether you want to use public key authentication. The values are not case sensitive. If you do not specify a value, false is assumed. You can omit this property when the operation target device is running Windows. **true** Specify this option to use public key authentication. **false** Specify this option to use password or keyboard interactive authentication. The default value is false.	Input

Property key	Property name	Description	I/O type
keyboardInteractiveAuthentication	SSH keyboard interactive authentication settings	Controls whether to use SSH keyboard interactive authentication for the Linux OS environment. If the OS of the destination is Linux, the system toggles between using and not using keyboard interactive authentication. If the property is set to true, keyboard interactive authentication is used. If the property is set to false, keyboard interactive authentication is not used. This property is not case-sensitive. This property is valid only when publicKeyAuthentication is set to false. If this property does not exist (which is true for a previous plug-in version) or if no value is specified, false is assumed for the property. The default value is false.	Input
elevatePrivileges	Elevate Privileges	If the OS of the operation target device is Linux, specify either of the following depending on whether you want to elevate the user to root privileges. The values are not case sensitive. If you do not specify a value, true is assumed. You can omit this property when the operation target device is running Windows. **true** Specify this option to run commands as a user with root privileges. **false** Specify this option to run commands without elevating the user to root. Commands are run	Input

Property key	Property name	Description	I/O type
		with the privileges of the connection user. The default value is false.	
transferMode R	Transfer Mode	Specify either of the following as the transfer mode: **send** Specify this option when transferring a file or folder from the Ops Center Automator server to the operation target device. When you specify a file path in the localFilePath property, the same path must be specified in the remoteFilePath property. When transferring a single file, if you specify different file names in the localFilePath and remoteFilePath properties, the file name specified in the remoteFilePath property applies. **receive** Specify this option when transferring a file or folder from the operation target device to the Ops Center Automator server. When you specify a file path in the remoteFilePath property, the same path must be specified in the localFilePath property. When transferring a single file, if you specify different file names in the remoteFilePath and localFilePath properties, the file name specified in the localFilePath property applies. The default value is send.	Input

Property key	Property name	Description	I/O type
localFilePath [R]	Local File Path	Specify the absolute path of the file or folder on the Ops Center Automator server using no more than 256 characters. In the localFilePath property, specify characters that can be used in commands in the OSs of the Ops Center Automator server and the operation target device. If there is a file or folder with the same name in the destination folder, the file or folder is overwritten. For this reason, you should specify a unique name. If the destination folder does not exist, the folder will be created in the specified configuration.	Input
remoteFilePath [R]	Remote File Path	Specify the absolute path of the file or folder on the operation target host in no more than 256 characters. In the remoteFilePath property, specify characters that can be used in commands in the OSs of the Ops Center Automator server and the operation target device. If the OS of the operation target device is Linux, make sure that the names of the files and folders you are transferring are encoded in the same character set used by the connection user. If there is a file or folder with the same name in the destination folder, the file or folder is overwritten. For this reason, you should specify a unique name. If the destination folder does not exist, the folder will be created in the specified configuration.	Input
[R]: Required			

When specifying file paths, use characters that can be used in commands in the OSs of the Ops Center Automator server and the operation target device. When specifying a file name in the localFilePath property, also specify a file name in the remoteFilePath property. When specifying a folder name in the localFilePath property, also specify a folder name in the remoteFilePath property.

Restrictions apply to the files and folders you can specify in the localFilePath and remoteFilePath properties.

If the operation target device is running Windows and a file with the Windows file attribute "Encrypt contents to secure data" is included among the transferred files, the transfer of the file fails, causing an error in the processing of the plug-in.

Restrictions on the names of transferred files and folders

The following table lists the restrictions that apply to the names of transferred files and folders when the connection destination is Windows or Linux OS:

Table 8 Sending

File or folder	Ops Center Automator side or destination host side	Property	Restrictions
File	Ops Center Automator	localFilePath	File names can be a maximum of 127 characters.
	Destination host	remoteFilePath	File names can be a maximum of 127 characters.
Folder	Ops Center Automator	localFilePath	The longest absolute path of the file or folder in the transferred folder can contain no more than 256 characters. The longest path from the folder being transferred to a file or folder under that folder must be no longer than 127 characters.
	Destination host	remoteFilePath	The longest absolute path of the file or folder in the transferred folder can contain no more than 256 characters. The longest path from the folder being transferred to a file or folder under that folder must be no longer than 127 characters.

Table 9 Receiving

File or folder	Ops Center Automator side or destination host side	Property	Restrictions
File	Ops Center Automator	localFilePath	File names can be a maximum of 127 characters.
	Destination host	remoteFilePath	File names can be a maximum of 127 characters.
Folder	Ops Center Automator	localFilePath	The longest absolute path of the file or folder in the transferred folder can contain no more than 256 characters. The longest path from the folder being transferred to a file or folder under that folder must be no longer than 127 characters.
	Destination host	remoteFilePath	The longest absolute path of the file or folder in the transferred folder can contain no more than 256 characters. The longest path from the folder being transferred to a file or folder under that folder must be no longer than 127 characters.

Specifying the SSH port number

You can specify a port number when using SSH to connect to the operation target device. The following table describes how to specify the port number and the priority of each method.

Priority	Set in	Property key	Default value
1	Connection-destination properties file (`connectiondestination-name.properties`)	ssh.port	--
2	Properties file (`config_user.properties`)	ssh.port.number	22

Specifying the SSH file transfer protocol

You can specify whether to use SFTP when using SSH to connect to the operation target device. If SFTP is not used, SCP is used.

The following table describes how to specify and the priority of each method.

Priority	Set in	Property key	Default value
1	Connection-destination properties file (`connectiondestination-name.properties`)	sftp.enable	--
2	Properties file (`config_user.properties`)	plugin.sftp.enable	false

Handling of forwarded files

Forwarded files are handled differently depending on the OS of the operation target device and the value specified in the transferMode property. The following table describes how forwarded files are handled:

Table 10 Windows

Item			send	receive
Time stamp of forwarded file	When creating a file	Creation date and time	Date and time of forwarding	Date and time of forwarding
		Update date and time	Update date and time of source file	Update date and time of source file
		Access date and time	Date and time of forwarding	Date and time of forwarding
	When overwriting a file	Creation date and time	Creation date and time of overwritten file	Creation date and time of overwritten file
		Update date and time	Update date and time of source file	Update date and time of source file
		Access date and time	Access date and time of overwritten file	Access date and time of overwritten file
Access permissions required for source file			System account read privilege	System account read privilege
Access permissions required for parent folder of destination file			Write privilege of the user set in the authentication information	System account write privilege

Item		send	receive
Access permissions required for destination file when overwriting the file		Write privilege of the user set in the authentication information	System account write privilege
Access permission assigned to destination file	When creating a file	Inherits privilege of parent folder	Inherits privilege of parent folder
	When overwriting a file	Inherits privilege of overwritten file	Inherits privilege of overwritten file

Table 11 Linux

Item			send	receive
Time stamp of forwarded file	When creating a file	Creation date and time	Date and time of forwarding	Date and time of forwarding
		Update date and time	Date and time of forwarding	Date and time of forwarding
		Access date and time	Date and time of forwarding	Date and time of forwarding
	When overwriting a file	Creation date and time	Date and time of forwarding	Creation date and time of overwritten file
		Update date and time	Date and time of forwarding	Date and time of forwarding
		Access date and time	Access date and time of overwritten file	Access date and time of overwritten file
Access permissions required for source file			System account read privilege	Read privilege of connection user
Access permissions required for parent folder of destination file			Write privilege of connection user	System account write privilege
Access permissions required for destination file when overwriting the file			Write privilege of connection user	System account write privilege

Item		send	receive
Access permission assigned to destination file	When creating a file	Uses the umask value of root or the connection user	Inherits privilege of parent folder
	When overwriting a file	Inherits privilege of overwritten file	Inherits privilege of overwritten file

Repeated Execution Plug-in

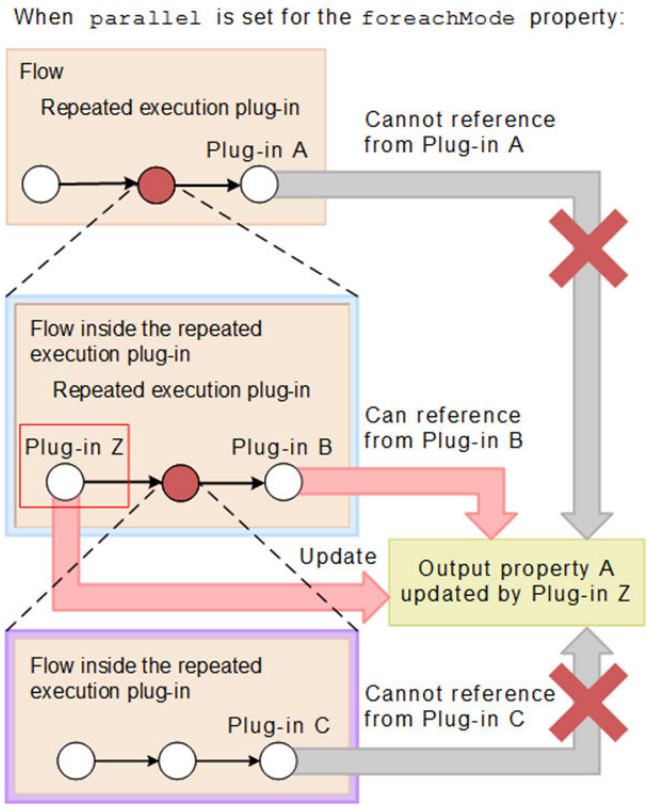
The Repeated Execution plug-in runs a specific flow repeatedly.

You can run a service with a value specified in the Input Properties (inputProperties) for each iteration of the flow. This is useful, for example, when you want to run the same processing on different servers. Note that the execution methods for a flow include concurrent execution that runs flows in parallel, and sequential execution that runs the next flow when the current flow finishes. In addition, the Repeated Execution Plug-in can have a nested structure, and it is possible to define the nest of Repeated Execution Plug-in up to 3 levels. When used in the combination with the Flow Plug-in, up to 25 levels can be defined together with the Flow Plug-in.

Cautionary notes

- If the parallel option is specified in the **`foreachMode`** property, the value of the properties (service properties, plug-in output properties, variables) referenced or updated in the context of a repeated task are only valid for the same repeated task (the nth flow). Additionally, the value of the property (service property, plug-in output property, variable) cannot be shared with concurrently processed, repeated tasks (except for the nth flow) or shared with the repeated tasks of a lower level.

The following figure shows how the plug-ins in each level reference an output property updated by plug-in Z.



Restrictions on the number of flows

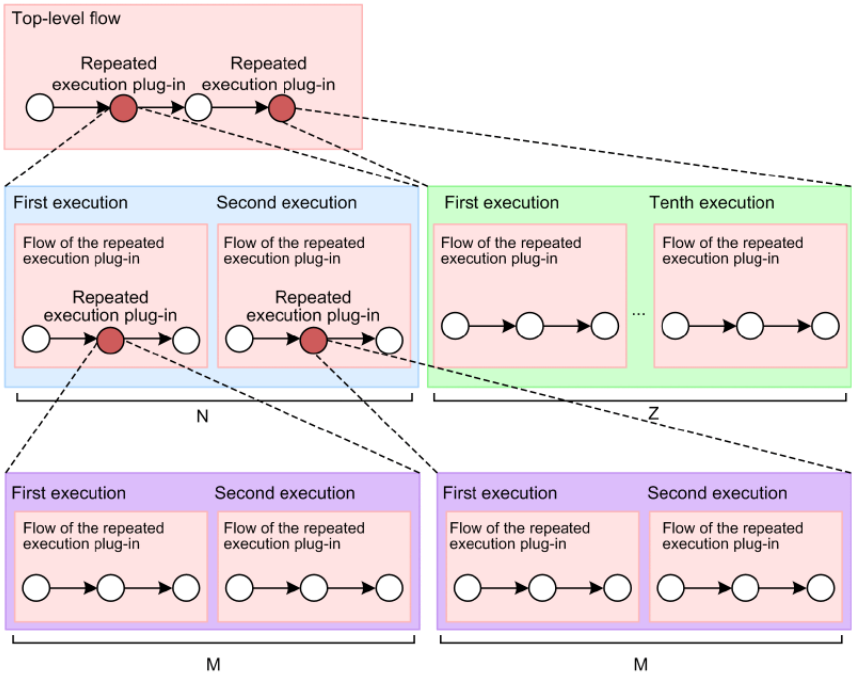
The maximum number of flows that can be specified for a Repeated Execution Plug-in is set through the **`inputProperties`** property and cannot exceed 10,000 per service. Note that the total number of flows does not include the number of flows run in the top-level flow or the flows associated with the Flow Plug-in.

If the number of flows under repetition in the service increases, the response might be delayed, and it might even cause the browser to crash. Therefore, the upper limit is set for the number of flows under repetition in the service.

The flow included in the count is only for the Repeated Execution Plug-in, while the flow for the Flow Plug-in is excluded from the counting target.

Type	Counting Target
Flow under Repeated Execution Plug-in	Yes
Flow by Flow Plug-in	Yes
Other flows	No

In the following example, the **`inputProperties`** properties of the Repeated Execution Plug-in is set to repeat plug-in N twice, plug-in M twice, and plug-in Z a total of ten times. Thus, the total number of flows is calculated as follows: N+N*M+Z (2+2*2+10) = 16.



The input values and loop index for the Repeated Execution plug-in a nested configuration that derives the input values and loop index values using the following reserved properties:

- reserved.loop.inputN
- reserved.loop.indexN

Note: If the Repeated Execution Plug-in is nested, you might encounter browser problems during debugging, in which case, you should reduce the value set for the property " inputProperties " for each Repeated Execution Plug-in, and then repeat the debugging.

Note: Set the property "inputProperties" for each Repeated Execution Plug-in so that the total number of flows does not exceed 10000. The flows for the Flow Plug-in are not included in the number of flows calculation.

Return Codes

The Repeated Execution Plug-in generates the following return codes:

Return Code	Description
0	Ended normally.
1	Some of the repeated processing failed.
2	All of the repeated processing failed.
3	The total number of flows under the Repeated Execution Plug-in in the service exceeds the upper limit.

Return Code	Description
65	The connection with the Ops Center Automator server failed. For example, the Ops Center Automator server might have stopped while the plug-in was being run.
66	The following user is mapped to the Ops Center Automator user: ▪ A user who does not belong to the Administrators group. ▪ A user other than the built-in Administrator who belongs to the Administrators group, in an environment with UAC enabled.
68	No information about the target job execution ID exists.
69	An environment variable of the task-processing engine cannot be acquired.
80	Task execution has stopped.
81	The plug-in was called in a status that is not valid.
82	The request message from the task-processing engine cannot be correctly parsed.
83	The environment of the Ops Center Automator server is corrupted.
84	Information about the specified plug-in cannot be obtained.
86	The specified property value is not valid.
127	Another error has occurred.

Property list

Following are the properties for the Repeated Execution Plug-in:

Property key	Property name	Description	I/O type
inputProperties [R]	Input Properties	Specifies an input property value for each repetition of the flow, using no more than 1,024 characters. You can specify a different property for each repetition. Use a comma to separate properties. Commas can only be used as delimiting characters. The maximum number of repetitions is 99. You cannot specify 100 or more comma separated values.	Input

Property key	Property name	Description	I/O type
outputProperties	Output Properties	Outputs the value of the output property for the number of repetitions. The total output is 1,024 characters or less. At each repetition, one property value is output separated by a comma in the order specified in the inputProperties property. Use a comma as the delimiting character.	Output
outputResult	Results	The execution result of each flow is output, separated by commas. **true** Output when the flow is run successfully. **false** Output when execution of the flow fails.	Output
foreachMode R	Mode	Specify the execution method for the repeated flow. **parallel** Repeated flows are run in parallel. A maximum of 99 flows can be run in parallel. If the maximum is exceeded, the excess flows are run when the number of executing flows falls below the maximum. You can change the number of concurrently executable flows between 1 and 99 using the foreach.max_value key in the properties file (`config_user.properties`). Even if an error occurs, all the unexecuted flows will be run. **serial** Repeated flows are run sequentially. If an error occurs, unexecuted repeated flows are not run.	Input

Property key	Property name	Description	I/O type
		The default value is parallel.	
R: Required			

Email Notification Plug-in

The Email notification plug-in sends emails to the specified destination.

This plug-in enables the connection to the SMTP server to transmit email with the specified recipient, subject, and body.

Prerequisites

The following information is obtained from built-in service share properties. Therefore, set the values for these items in advance in the System Settings view.

- Address of the SMTP server
- Port number
- User ID
- Password
- Originator of the notification email

Cautionary notes

- Even if you do not specify the toAddress, ccAddress, and bccAddress properties, the return code will be 0.
- The mail address to be specified differs from the value of the built-in service share properties. Therefore, make sure that you specify at least one of the toAddress, ccAddress, and bccAddress properties.
- If any of the toAddress, ccAddress, and bccAddress properties has an email address specified that is not valid, email transmission will fail to all the addresses.
- If you use machine-dependent characters or characters that are incompatible between character sets in the mailSubject or mailBody property, the characters are replaced with question marks (?) or other characters. In this scenario, either change the characters in the email, or change the encoding.
- The following characters might not be converted correctly:

  ~, \, ＼, ～, ‖, －, ¢, £, ¬
- If the execution of a task is stopped while the plug-in is executing, the status of the task becomes Failed or Completed when the processing of the email notification plug-in finishes. The status of steps and tasks after plug-in execution has finished depends on the return code of the step and the condition for executing subsequent steps. You can set a Subsequent-step Execution Condition in the Create Step dialog box or the Edit Step dialog box.

Return codes

The Email Notification Plug-in generates the following return codes:

Return Code	Description
0	Ended normally.
65	The connection with the Ops Center Automator server failed. For example, the Ops Center Automator server might have stopped while the plug-in was being run.
66	The following user is mapped to the Ops Center Automator user: ▪ A user who does not belong to the Administrators group. ▪ A user other than the built-in Administrator who belongs to the Administrators group, in an environment with UAC enabled.
68	No information about the target job execution ID exists.
69	An environment variable of the task-processing engine cannot be acquired.
70	The connection with the SMTP server failed.
78	Authentication failed.
79	Email transmission failed.
80	Task execution has stopped.
81	The plug-in was called in a status that is not valid.
82	The request message from the task-processing engine cannot be correctly parsed.
83	The environment of the Ops Center Automator server is corrupted.
84	Information about the specified plug-in cannot be obtained.
86	The specified property value is not valid.
127	Another error has occurred.

Property list

The following properties are available for the Email Notification Plug-in:

Property key	Property name	Description	Default value	I/O type	Required
toAddress	To Addresses	Specify the email addresses of recipients to enter in the TO attribute, using no more than 1,024 characters. When specifying multiple addresses, separate them with commas.	--	Input	O
ccAddress	Cc Addresses	Specify the email addresses of recipients to enter in the CC attribute, using no more than 1,024 characters. When specifying multiple addresses, separate them with commas.	--	Input	O
bccAddress	Bcc Addresses	Specify the email addresses of recipients to enter in the BCC attribute, using no more than 1,024 characters. When specifying multiple addresses, separate them with commas.	--	Input	O
encodeType	Encoding	Specify the encoding of the email as one of the following: ▪ us-ascii ▪ iso-2022-jp ▪ shift_jis ▪ euc-jp ▪ utf-8	utf-8	Input	R
mailSubject	Subject	Specify the subject line of the email using no more than 256 characters.	--	Input	R
mailBody	Body	Specify the body text of the email using no more than 1,024 characters.	--	Input	O

User-Response Wait Plug-in

The User-Response Wait Plug-in enables a user to select the processing of the succeeding step when running the service.

To select the processing, the user uses the **Respond** dialog box. You can also set up email notification to notify the user that a task is waiting for a response.

You can access the **Respond** dialog box as follows:

- Link from the URL in the response wait notification mail
- Link from the Tasks view

Prerequisites

The following information is obtained from built-in service share properties. Therefore, to notify the user when a task is waiting for a response, set the values for these parameters in advance in the System Settings view.

- Address of the SMTP server
- Port number
- User ID
- Password
- Originator of the notification email

Cautionary notes

- The email reporting that a task is waiting for a user response is not sent if any of the following applies:
  - No value is set in the built-in service share property.
  - SMTP has not been set up.
  - None of the toAddress, ccAddress, and bccAddress properties are specified.
  - An email address that is not valid is specified in any of the toAddress, ccAddress, and bccAddress properties.
- The mail address to be specified differs from the value of the built-in service share properties. Therefore, make sure that you specify at least one of the toAddress, ccAddress, and bccAddress properties.
- Do not stop the user-response wait plug-in while the **Respond** dialog box is visible and waiting for a response. Stopping it causes an error even if the operator selects processing for the subsequent step.
- A URL that links to the **Respond** dialog box is automatically entered in the body of the notification email. If more than one step in a task is waiting for a response, each step that runs the user-response wait plug-in has a different URL, with each URL visible in the **Respond** dialog box for that step.
- When you respond in other than the URL that links to the **Respond** dialog box, if there are multiple responses waiting, the response must be input from the oldest waiting response.

- You cannot change the layout of the **Respond** dialog box.
- Any return code from the properties labelButton1 to labelButton9 is considered an abnormal end, and error information is output to the task log. For the labelButton0 property and the properties labelButton1 to labelButton9, if the output log level is 10 or 20, the output details in the task log differ depending on the response result.
- If you use machine-dependent characters or characters that are incompatible between character sets in the mailSubject or mailBody property, the characters are replaced with question marks (?) or other characters. In this scenario, either change the characters in the email, or change the encoding.
- The following characters might not be converted correctly:

  ~, \, ＼, ～, ‖, －, ￠, ￡, ￢
- Even if the task is stopped or forcibly stopped, when you run Retry the Task From the Step After the Failed Step, the subsequent steps are run without response input. To prevent subsequent steps from being run, we recommended that you uncheck Retry under [Available Actions] in the **Create/Edit Service** window.
- When specifying users who can respond with the responseUser property, note the following:
  - If a task is run by specifying only non-existent users or the user who submitted the task for the responseUser property, the task cannot be responded to using any user. In this case, you must stop or forcibly stop the task.
  - You cannot respond as the user who submitted the task, even in the Service Builder Debug window. To run subsequent plug-ins of User-Response Wait Plug-in on the Service Builder Debug window, either do not specify a value for the responseUser property or run User-Response Wait Plug-in in dry-run mode.

Return codes

The User-Response Wait Plug-in generates the following return codes:

Return Code	Description
0-9	Returns the return code corresponding to the labelButton1 to labelButton9 properties. If a timeout occurs while waiting for a response, the value specified in the timeOutDefault property is returned as the return code. Therefore, the meaning of the return code depends on the service template that is using the plug-in.
10-63	If a timeout occurs while waiting for a response, the value specified in the timeOutDefault property is returned as the return code.
65	The connection with the Ops Center Automator server failed. For example, the Ops Center Automator server might have stopped while the plug-in was being run.

Return Code	Description
66	The following user is mapped to the Ops Center Automator user: ▪ A user who does not belong to the Administrators group. ▪ A user other than the built-in Administrator who belongs to the Administrators group, in an environment with UAC enabled.
68	No information about the target job execution ID exists.
69	An environment variable of the task-processing engine cannot be acquired.
80	Task execution has stopped.
81	The plug-in was called in a status that is not valid.
82	The request message from the task-processing engine cannot be correctly parsed.
83	The environment of the Ops Center Automator server is corrupted.
84	Information about the specified plug-in cannot be obtained.
86	The value set for the mapping parameter in the **Response Input** dialog box is contrary to the property restriction.
127	Another error has occurred.

Property list

The following properties are available from the User-Response Wait Plug-in:

Property key	Property name	Description	Default value	I/O type	Required
toAddress	To Addresses	Specify the email addresses of recipients to enter in the TO attribute, using no more than 1,024 characters. When specifying multiple addresses, separate them with commas.	--	Input	false
ccAddress	Cc Addresses	Specify the email addresses of recipients to enter in the CC attribute, using no more than 1,024 characters. When specifying multiple addresses, separate them with commas.	--	Input	false

Property key	Property name	Description	Default value	I/O type	Required
bccAddress	Bcc Addresses	Specify the email addresses of recipients to enter in the BCC attribute, using no more than 1,024 characters. When specifying multiple addresses, separate them with commas.	--	Input	false
mailSubject	Subject	Specify the subject line of the email using no more than 256 characters.	--	Input	false
mailBody	Body	Specify the body text of the email using no more than 1,024 characters.	--	Input	false
encodeType	Encoding	Specify the encoding of the email as one of the following: ▪ us-ascii ▪ iso-2022-jp ▪ shift_jis ▪ euc-jp ▪ utf-8	utf-8	Input	false
dialogText	Response Input Dialog Box	Specify the information to display in the Respond dialog box. You can specify the information in text or HTML format.	--	Input	true
responseTimeOut	Response Timeout	Specify the time, between 1 and 20,160 (in minutes), before timeout occurs while waiting for a response.	1440	Input	true

Property key	Property name	Description	Default value	I/O type	Required
timeOutDefault	Default Return Value	Specify the return code to return when a timeout occurs while waiting for a response. When the timeout period has passed, this value is returned as the return code. For example, when 0 is specified and the timeout period elapses, the processing corresponding to the option associated with the labelButton0 property will be run. Specify a return code in the range from 0 to 63.	0	Input	true
labelButton0	Label Button 0	Specify the option label for the response that generates return code 0, using a maximum of 256 characters. You can display options that meet the user's functional needs in the Respond dialog box.	--	Input	false
labelButton1	Label Button 1	Specify the option label for the response that generates return code 1, using a maximum of 256 characters. You can display options that meet the user's functional needs in the Respond dialog box. If you omit this property, the corresponding option is not visible.	--	Input	false
labelButton2	Label Button 2	Specify the option label for the response that generates return code 2, using a maximum of 256 characters. you can display options that meet the user's functional needs in the Respond dialog box. If you omit this property, the corresponding option is not visible.	--	Input	false

Property key	Property name	Description	Default value	I/O type	Required
labelButton3	Label Button 3	Specify the option label for the response that generates return code 3, using a maximum of 256 characters. you can display options that meet the user's functional needs in the Respond dialog box. If you omit this property, the corresponding option is not visible.	--	Input	false
labelButton4	Label Button 4	Specify the option label for the response that generates return code 4, using a maximum of 256 characters. you can display options that meet the user's functional needs in the Respond dialog box. If you omit this property, the corresponding option is not visible.	--	Input	false
labelButton5	Label Button 5	Specify the option label for the response that generates return code 5, using a maximum of 256 characters. you can display options that meet the user's functional needs in the Respond dialog box. If you omit this property, the corresponding option is not visible.	--	Input	false
labelButton6	Label Button 6	Specify the option label for the response that generates return code 6, using a maximum of 256 characters. you can display options that meet the user's functional needs in the Respond dialog box. If you omit this property, the corresponding option is not visible.	--	Input	false

Property key	Property name	Description	Default value	I/O type	Required
labelButton7	Label Button 7	Specify the option label for the response that generates return code 7, using a maximum of 256 characters. you can display options that meet the user's functional needs in the Respond dialog box. If you omit this property, the corresponding option is not visible.	--	Input	false
labelButton8	Label Button 8	Specify the option label for the response that generates return code 8, using a maximum of 256 characters. you can display options that meet the user's functional needs in the Respond dialog box. If you omit this property, the corresponding option is not visible.	--	Input	false
labelButton9	Label Button 9	Specify the option label for the response that generates return code 9, using a maximum of 256 characters. you can display options that meet the user's functional needs in the Respond dialog box. If you omit this property, the corresponding option is not visible.	--	Input	false

Property key	Property name	Description	Default value	I/O type	Required
responseUser	Users who can respond	Specify the users who can respond[1], using a maximum of 1024 characters. The specifiable characters are half-width alphanumeric characters and symbols (! # $ % & ' ( ) * + - . = @ \ ^ _ |). When specifying multiple users, separate them with commas. The user who submitted the service cannot respond even if that user is specified. If no users are entered, then all users can respond, including the user who submitted the service.[2]	--	Input	false

1. When specifying the users authenticated by Common Services, specify the users in the following format:

Authentication type	What to specify
User directory (Active Directory)	Specify the User ID at login.
User directory (LDAP server)	
Local users	
ID provider (OIDC)	Specify the User ID in UPN format.
ID provider (SAML)	Specify the User ID in the Claim Issuance Policy setting for AD FS and NameID Policy Format in Common Services.

For details, see the *Hitachi Ops Center Installation and Configuration Guide*.

When specifying the users authenticated by Common Component, specify the User ID at login.

Property key	Property name	Description	Default value	I/O type	Required

Note: When responding by the users authenticated by Common Component, be sure to log in with the value specified for the responseUser property. When specifying a user registered in an external authentication server or external authorization server, if the login format is different from the format specified for the responseUser property, such as User ID including the domain name or User ID not including the domain name, response input cannot be done.

2. If the minimum length of the responseUser property is set to 1 or more, it cannot be set to empty (default value). When using this property, it is recommended that the minimum length of the property be set to 1 or more in order to make the setting of responseUser mandatory in the **Create/Edit Service** and **Submit Service Request** windows. See Specify the property settings (on page 50) for details.

HTML tags and attributes that can be specified in the dialogText property

When specifying the display contents in the dialogText property in HTML format, use the tags listed in the following table:

Tag	Attribute	Notes and restrictions
Anchor tag (<a>)		When using the <a> tag, the target attribute must be left blank. If this tag is not specified, a page is loaded into the **Response Input** dialog box.
	href	-
	target	Specify "_blank."
Bold tag (<b>)		-
Break tag ( )		-
Font tag (<font>)	color	-
	face	-
	size	-
Italic tag (<i>)		-
Underline tag (<u>)		-
Form tag (<form>)		-

Tag	Attribute	Notes and restrictions
Input tag (<input>)		It is common to be enclosed with form tags, but since it is not required to be sent, there is no need to be enclosed with form tags.
	name	If you specify the service property key, you can change the mapping parameter of the service property specified in the **Response Input** dialog box while waiting for a response.
	type	"text", "check box", "radio" can be specified.
	value	When the type attribute is "check box", "radio", set the value attribute. The value set in the value attribute of the item checked with the check box or the radio control element becomes the mapping parameter of the service property specified by the name attribute.
Select tag (<select>)	name	If you specify the service property key, you can change the mapping parameter of the service property specified in the **Response Input** dialog box while waiting for a response.
Option tag (<option>)		Use select tags to enclose.
	value	The value set in the value attribute of the item selected in the select menu or the list box becomes the mapping parameter of the service property specified by the name attribute of the select tag.

Terminal Connect Plug-in

Enables terminals to establish connections.

The Terminal Connect Plug-in allows you to connect to an operation target device by using Telnet or SSH and authenticate.

When connecting by Telnet, set the user ID and password as needed. For SSH connections, you can select password authentication, public key authentication, or keyboard interactive authentication as the authentication method. You must set the following information in the plug-in properties or from the Agentless Remote Connections view.

- Authentication method (password authentication, public key authentication or keyboard interactive authentication)
- Information required for password authentication (user ID and password)
- Information required for public key authentication (user ID)
- Information required for keyboard interactive authentication (user ID and password)

The commands specified in the terminal command plug-in are run with the privileges of the user authenticated by the terminal connect plug-in. To run a command with root privileges, you must run the command in the terminal command plug-in that elevates the user to root privileges.

Prerequisites

- The plug-in uses the protocol specified in the protocol property to communicate with the Ops Center Automator server.
- When connecting by Telnet, the plug-in detects when the operation target device is prompting the operator for a user ID and password. Set one of the following files as needed. If you set both files, Ops Center Automator uses the values set in the connection-destination properties file (`connection-destination-name.properties`).
  - telnet.prompt.account and telnet.prompt.password in the connection-destination properties file (`connection-destination-name.properties`)
  - plugin.terminal.prompt.account and plugin.terminal.prompt.password in the properties file (`config_user.properties`)

Cautionary notes

- The plug-in waits for standard output for the length of time specified in the readWaitTime property. If the time specified in readWaitTime elapses after output to standard output has ceased, plug-in execution ends in an error. Make sure that the value of the readWaitTime property is valid before using the plug-in.
- If the value output to standard output matches the regular expression pattern specified in the promptPattern property, the plug-in ends immediately.

- After using Telnet to establish a connection to an operation target device, the plug-in waits for standard output and standard error output for the length of time set in the telnet.connect.wait property in the properties file (`config_user.properties`). If the connection destination service is a Web server or other entity that does not produce standard output or standard error output, set the port number of the service in the telnet.noStdout.port.list property of the connection-destination properties file (`connection-destinationname.properties`). If you set the port number, the plug-in finishes executing without waiting for standard output or standard error output.
- If the execution of a task is stopped while the plug-in is executing, the status of the task becomes Failed or Completed when the processing of the terminal connect plug-in finishes. The session and token are then discarded. The status of steps and tasks after plug-in execution has finished depends on the return code of the step and the condition for executing subsequent steps. You can set a Subsequent-step Execution Condition in the Create Step dialog box or the Edit Step dialog box.
- The terminal connect plug-in maintains the connection even if Telnet authentication fails. To end the connection, you must run a terminal disconnect plug-in. However, if the task enters Failed or Completed status, the connection ends automatically and you do not must run the terminal disconnect plug-in.
- The standard output and standard error output of a terminal connect plug-in is output as the standard output of the Ops Center Automator step. The size of the standard output and standard error output is the total number of bytes received by Ops Center Automator. If the Telnet server or SSH server is configured to replace the linefeed character LF with CR+LF, allow two bytes for each linefeed character. The results of processing whose total standard output and standard error output exceeds 100 KB is outside the scope of product support. Make sure that the total standard output and standard error output does not exceed 100 KB.
- The terminal connect plug-in cannot detect authentication errors in Telnet connections. For this reason, specify a regular expression pattern that detects authentication errors in standard output and standard error output in any of stdoutPattern1 to stdoutPattern3.
- When the Terminal Connect Plug-in version is less than 02.00.00, "patternMatch" is set in "outputCondition". But, the default value of "outputCondition" is "always" in the case of version 02.00.00. When you upgrade Terminal Connect Plug-in, keep this in mind.

Return codes

The Terminal Connect Plug-in generates the following return codes:

Return Code	Description
0 - 63	If standard output or standard error output matches the regular expression pattern specified in the returnCodePattern property, the plug-in returns the return code specified in the returnCode property. If standard output and standard error output do not match the pattern specified in the returnCodePattern property, the plug-in returns the return code specified in the defaultReturnCode property. Therefore, the meaning of the return code depends on the service template that is using the plug-in.

Return Code	Description
65	The connection with the Ops Center Automator server failed. For example, the Ops Center Automator server might have stopped while the plug-in was being run.
66	The following user is mapped to the Ops Center Automator user: ▪ A user who does not belong to the Administrators group. ▪ A user other than the built-in Administrator who belongs to the Administrators group, in an environment with UAC enabled.
68	No information about the target job execution ID exists.
69	An environment variable of the task-processing engine cannot be acquired.
70	The connection with the operation target device failed.
76	The connection timed out.
77	The host name of the operation target device cannot be resolved.
78	When connecting by SSH, authentication on the operation target device failed.
80	Task execution has stopped.
81	The plug-in was called in a status that is not valid.
82	The request message from the task-processing engine cannot be correctly parsed.
83	The environment of the Ops Center Automator server is corrupted.
84	Information about the specified plug-in cannot be obtained.
86	The specified property value is not valid.
87	Standard output and standard error output have timed out.
88	The maximum number of tokens (99 per task) has been reached. The total standard output and standard error output has exceeded 100 KB.
127	Another error has occurred.

Property list

The following properties are available for the Terminal Connect Plug-in:

Property key	Property name	Description	I/O type	Required
destinationHost	Destination Host	Specify the IPv4 address, IPv6 address, or host name of the operation target device using no more than 1,024 characters. Multiple IP addresses or host names cannot be specified.	Input	true
protocol	Protocol	Specify the protocol to use when connection to the operating target device. You can specify the following protocols: ▪ Telnet ▪ SSH The default is Telnet.	Input	false
credentialType	Credentials Type	As the authentication type to use during command or script execution, specify either of the following: **Destination** Specify this option to use the authentication information set in the Agentless Remote Connections view. Specifying destination applies the authentication information set for Telnet or SSH in the connection destination definition according to the IP address of the Ops Center Automator login user. You can omit the specification of properties relating to authentication information (account, password, suPassword, publicKeyAuthentication, and	Input	true

Property key	Property name	Description	I/O type	Required
		keyboardInteractiveAuthentication). **Property** Specify this option to use the values specified in the following properties as authentication information: ▪ account ▪ password ▪ suPassword ▪ publicKeyAuthentication ▪ keyboardInteractiveAuthentication		
account	User ID	Specify the user ID to use to log on to the operation target device, using a maximum of 256 characters. This property is required if both of the following are true: ▪ SSH is specified in the protocol property. ▪ property is specified in the credentialType property.	Input	false
password	Password	Specify the password to use to log on to the operation target device, using a maximum of 256 characters. This property is required if all of the	Input	false

Property key	Property name	Description	I/O type	Required
		following conditions are met: ▪ SSH is specified in the protocol property. ▪ property is specified in the credentialType property. ▪ false is specified in the publicKeyAuthentication property. If the OS of the operation target device is Linux and true is specified for the publicKeyAuthentication property, any value you specify is ignored. You can, however, set a value for the reserved.terminal.password reserved property to reference.		
suPassword	Administrator Password	Specify the password required to elevate the user to root privilege, using a maximum of 256 characters. The value of the suPassword property is assigned to the reserved.terminal.suPassword property when you specify the latter in the command line or a terminal command plug-in.	Input	false
publicKeyAuthentication	SSH public key authentication settings	If the OS of the operation target device is Linux, specify either of the following depending on whether you want to use public key authentication. The values are not case sensitive. If you do not specify a value, false is assumed. You can omit	Input	false

Property key	Property name	Description	I/O type	Required
		this property when the operation target device is running Windows. **true** Specify this option to use public key authentication. **false** Specify this option to use password authentication or keyboard interactive authentication. The default is false.		
keyboardInteractiveAuthentication	SSH keyboard interactive authentication settings	Controls whether to use SSH keyboard interactive authentication for the Linux OS environment. If the OS of the destination is Linux OS, the system toggles between using and not using keyboard interactive authentication. If the property is set to true, keyboard interactive authentication is used. If the property is set to false, keyboard interactive authentication is not used. This property is not case-sensitive. This property is valid only when publicKeyAuthentication is set to false. If this property does not exist (which is true for a previous plug-in version) or if no value is specified, false is assumed for the property. The default is false.	Input	false
Port	Port Number	Specify the port number to use when connecting to the operating target device.	Input	false

Property key	Property name	Description	I/O type	Required
charset	Character Set	Specify the character set to use when writing to the standard input of the operation target device and reading from standard output and standard error output. Specify the same character set as that of the user who logs in to the operation target. The names of character sets are not case sensitive. You can specify the following character sets: ▪ EUC-JP ▪ eucjp ▪ ibm-943C ▪ ISO-8859-1 ▪ MS932 ▪ PCK ▪ Shift_JIS ▪ UTF-8 ▪ windows-31j	Input	false
lineEnd	Newline Character	When Telnet is specified in the protocol property of the terminal connect plug-in, specify the newline character to append to the values specified in the account and password properties. You can specify the following: ▪ CR ▪ LF ▪ CRLF	Input	false

Property key	Property name	Description	I/O type	Required
		To use 0x0D as the newline character, specify CR. To use 0x0A, specify LF, and to use 0x0D0A, specify CRLF. The default is CR.		
promptPattern	Prompt Pattern	Specify the regular expression pattern to use to detect prompts in standard output and standard error output, using no more than 1,024 characters. This property is used to detect when the operation target device is ready to run commands after the connection is established. Specify the pattern in a PCRE-compliant format. When the output matches the specified regular expression pattern, the plug-in ends immediately. If the output does not match the pattern, the plug-in ends in an error when the time set in the readWaitTime property has elapsed since the last output to standard output or standard error output.	Input	true
readWaitTime	Standard Output Wait Time	When logging in to an operation target device, specify how long to wait after output to standard output or standard error output until the next output. Specify the timeout time in a range from 1 to 86,400,000 (in milliseconds). The default is 60000	Input	false

Property key	Property name	Description	I/O type	Required
token	Token String	The token string that identifies the session is output to this property. You can specify the character string output to this property in the token property of terminal command plug-ins and terminal disconnect plug-ins.	Output	false
outputCondition	Output Condition	Specifies a condition to be output to the standard output property 1-3. You can specify the following values: ▪ always -- Outputs a null character even if it does not match the specified pattern. ▪ patternMatch -- Outputs only when matching the standard output pattern 1-3. If there is no output in output properties, mapped service properties are also not updated. The default is always.	Input	false
stdoutPattern1	Standard Output Pattern 1	Specify the regular expression pattern of the standard output and standard error output to output the stdoutProperty property, using a maximum of 1,024 characters. Specify the pattern in a PCRE-compliant format. If you use more than 1,024 characters, the 1,025th and subsequent characters are discarded.	Input	false

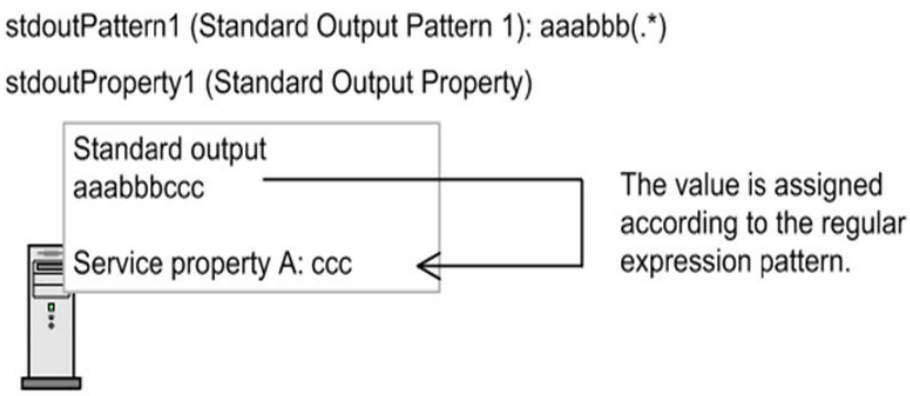
Property key	Property name	Description	I/O type	Required
stdoutProperty 1	Standard Output Property 1	The character string extracted by the stdoutPattern1 property is output to this property.	Output	false
stdoutPattern2	Standard Output Pattern 2	Specify the regular expression pattern of the standard output and standard error output to output the stdoutProperty property, using a maximum of 1,024 characters. Specify the pattern in a PCRE-compliant format. If you use more than 1,024 characters, the 1,025th and subsequent characters are discarded.	Input	false
stdoutProperty 2	Standard Output Property 2	The character string extracted by the stdoutPattern2 property is output to this property.	Output	false
stdoutPattern3	Standard Output Pattern 3	Specify the regular expression pattern of the standard output and standard error output to output the stdoutProperty property, using a maximum of 1,024 characters. Specify the pattern in a PCRE-compliant format. If you use more than 1,024 characters, the 1,025th and subsequent characters are discarded.	Input	false
stdoutProperty 3	Standard Output Property 3	The character string extracted by the stdoutPattern3 property is output to this property.	Output	false

Property key	Property name	Description	I/O type	Required
defaultReturnCode	Default Return Code	Specify the value to return as the return code when standard output and standard error output do not match the regular expression pattern specified in the returnCodePattern property. Specify a value in the range from 0 to 63. The default is 0.	Input	false
returnCodePattern	Return Code Pattern	Specify the regular expression pattern for standard output and standard error output, using a maximum of 1,024 characters. Specify the pattern in a PCRE compliant format. If standard output and standard error output match the specified pattern, the plug-in returns the value specified in the returnCode property.	Input	false
returnCode	Return Code	Specify the return code to be returned by the plug-in when standard output and standard error output match the pattern set in the returnCodePattern property. You can specify a value in the range from 0 to 63. If you omit this property, the plug-in returns the value specified in the defaultReturnCode property.	Input	false

Usage example of stdoutPattern and stdoutProperty properties

By using the stdoutPattern property, you can extract the value output to standard output and store it in the stdoutProperty property. The following figure shows the data flow when specifying aaabbb(.*) in stdoutPattern1.

Following is a usage example of stdoutPattern and stdoutProperty properties:



As defined in stdoutPattern1, for the standard output aaabbbccc, the value after aaabbb (in this case ccc) is extracted. The extracted value is stored in the stdoutProperty1 property.

Priority when plug-in properties are set in several locations

Information related to plug-in properties can also be set in a connection destination properties file (`connection destination-name.properties`) or the properties file (`config_user.properties`). When a value is set for a property in multiple locations, the following priority applies:

Setting	Location	Property key	Priority	Default value
Telnet port number	Plug-in property	port	1	--
	Connection destination properties file (`connection-destination-name.properties`)	telnet.port	2	--
	Properties file (`config_user.properties`)	telnet.port.number	3	23
SSH port number	Plug-in property	port	1	--
	Connection destination properties file (`connection-destination-name.properties`)	ssh.port	2	--

Setting	Location	Property key	Priority	Default value
	Properties file (`config_user.properties`)	ssh.port.number	3	22
Character set name	Plug-in property	charset	1	--
	Connection destination properties file (`connection-destination-name.properties`)	terminal.charset	2	--

If no value is set in the plug-in property or the connection destination properties file (connection-destination-name.properties), UTF-8 is set.

Usage examples of terminal connect plug-in

Example of judging Telnet authentication errors

The following describes an example of using plug-in properties to achieve the following processing:

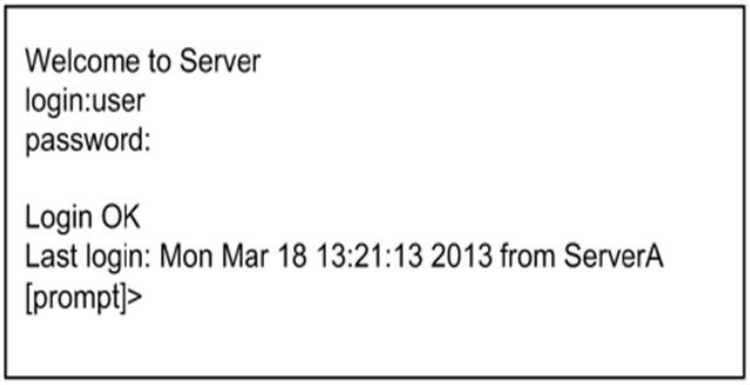
- Return 0 when login is successful.
- Return 1 when login fails.
- When login is successful, store the date and time of the last login and information about the connection source in the stdoutProperty1 property.

The following table describes examples of the values you can specify in plug-in properties to achieve this processing.

Property key	Example of specified value	Meaning of specified value
promptPattern	^\[prompt\]|^Login incorrect	If the contents of standard output matches [prompt] or Login incorrect, the plug-in ends and determines the return code.
stdoutPattern1	^Last login:(.*)	The character string following Last login: in standard output is stored in the stdoutProperty1 property.

Property key	Example of specified value	Meaning of specified value
defaultReturnCode	0	If the contents of standard output do not match the value specified in the returnCodePattern property, 0 is returned.
returnCodePattern	^ Login incorrect	If the contents of standard output match Login incorrect, the plug-in returns the return code specified in the returnCode property.
returnCode	1	If the contents of standard output matches the value specified in the returnCodePattern property, the plug-in returns 1.

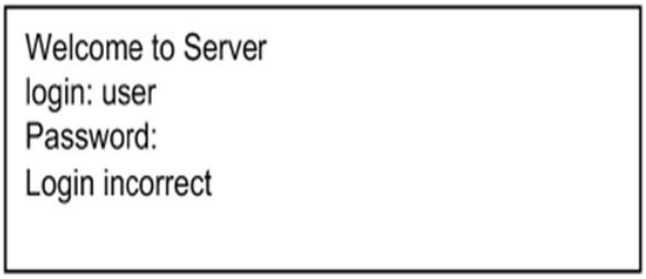
The following describes the function of a plug-in with the properties listed previously when it encounters the following standard output.



This is an example when the login is successful.

Because the contents of standard output match the value specified in the promptPattern property, the terminal connect plug-in determines the return code. In this case, because the standard output does not match the value specified in the returnCodePattern property, the plug-in returns code (0), the value specified in the defaultReturnCode property.

The character string extracted by the stdoutPattern1 property (Mon Mar 18 13:21:13 2013 from ServerA) is stored in the stdoutProperty1 property.



This is an example when the login fails.

Because the contents of standard output match the value specified in the promptPattern property, the return code of the terminal connect plug-in is determined. In this case, because the return code matches the value specified in the returnCodePattern property, the plug-in returns code (1), the value specified in the returnCode property.

Verifying whether an authentication error occurred when using SSH

When using SSH as the protocol, you can verify whether an authentication error has occurred by reviewing the return code of the terminal connect plug-in.

Authentication errors are detected using the authentication information set in the Agentless Remote Connections view or the authentication-related properties of the terminal connect plug-in (account, password, and publicKeyAuthentication). This process does not use the superuser password set in the Agentless Remote Connections view or the suPassword property of the terminal connect plug-in.

If an authentication error is detected, the plug-in returns code 78. Note that the return code of the plug-in will be 70 if destination is specified for the credentialType property and the authentication information in the Agentless Remote Connections view is set incorrectly.

Example of connecting to a service such as an HTTP server that does not produce standard output

The following describes an example of connecting to a service that does not produce standard output. This example assumes that 80 is specified in the telnet.noStdout.port.list property in the connection-destination properties file (`connection-destination-name.properties`).

In this case, the values specified in the following properties are ignored, and the plug-in returns code 0.

- credentialType
- account
- password
- suPassword
- publicKeyAuthentication
- keyboardInteractiveAuthentication
- charset
- lineEnd

- promptPattern
- readWaitTime
- stdoutPattern1 to stdoutPattern3
- defaultReturnCode
- returnCodePattern
- returnCode

Terminal Command Plug-in

The Terminal Command Plug-in runs commands on the destination host that was connected to by using the Terminal Connect Plug-in.

Function

This plug-in allows you to run a specified command on an operation target device that is connected by using the terminal connect plug-in. The commands specified in the terminal command plug-in are run with the privileges of the user authenticated by the terminal connect plug-in. To run a command with root privileges, the terminal command plug-in must run the command that elevates the user to root privileges.

Prerequisites

- The protocol specified in the protocol property of the terminal connect plug-in is used to communicate with the Ops Center Automator server.
- A connection must have been established with the operation target device by a terminal connect plug-in.

Cautionary notes

- The plug-in waits for standard output for the length of time specified in the readWaitTime property. If the time specified in readWaitTime elapses after output to standard output has ceased, plug-in execution ends in an error. Make sure that the value of the readWaitTime property is valid before using the plug-in. Any information output after the plug-in has timed out is discarded.
- If the value output to standard output matches the regular expression pattern specified in the promptPattern property, the plug-in ends immediately.
- If the command outputs information one page at a time, the system assumes that standard output has ceased. If the time specified in the readWaitTime property then passes, the plug-in ends with an error. Make sure that the command run by the terminal command plug-in is not configured to output results one page at a time.
- Echoed command lines are also output to standard output. When needed, configure the command to not echo back.

- If execution of a task is stopped during plug-in execution, the status of the task becomes Failed or Completed when the processing of the terminal command plug-in has finished. The session and token are then discarded. The status of steps and tasks after plug-in execution has finished depends on the return code of the step and the condition for executing subsequent steps. You can set Subsequent-step Execution Condition in the Create Step dialog box or the Edit Step dialog box.
- The standard output and standard error output of the terminal command plug-in is output as the standard output of the Ops Center Automator step. The size of the standard output and standard error output is the total number of bytes received by Ops Center Automator. If the Telnet server or SSH server is configured to replace the linefeed character LF with CR+LF, allow two bytes for each linefeed character. The results of processing whose total standard output and standard error output exceeds 100 KB is outside the scope of product support. Make sure that the total standard output and standard error output does not exceed 100 KB.
- If you intend to specify non-ASCII characters in the commandLine property, see General Command Plug-in.
- When the Terminal Connect Plug-in version is less than 02.00.00, "patternMatch" is set in "outputCondition". But, the default value of "outputCondition" is "always" in the case of version 02.00.00. When you upgrade Terminal Connect Plug-in, keep this in mind.

Return codes

The Terminal Command Plug-in generates the following return codes:

Return Code	Description
0-63	If standard output and standard error output match the regular expression pattern specified in the returnCodePattern property, the plug-in returns the return code specified in the returnCode property. However, if the output does not match the pattern, the plug-in returns the return code specified in the defaultReturnCode property. Therefore, the meaning of the return code depends on the service template that is using the plug-in.
65	The connection with the Ops Center Automator server failed. For example, the Ops Center Automator server might have stopped while the plug-in was being run.
66	The following user is mapped to the Ops Center Automator user: ▪ A user who does not belong to the Administrators group. ▪ A user other than the built-in Administrator who belongs to the Administrators group, in an environment with UAC enabled.
68	No information about the target job execution ID exists.
69	An environment variable of the task-processing engine cannot be acquired.
70	The connection with the operation target device failed.

Return Code	Description
80	Task execution has stopped.
81	The plug-in was called in a status that is not valid.
82	The request message from the task-processing engine cannot be correctly parsed.
83	The environment of the Ops Center Automator server is corrupted.
84	Information about the specified plug-in cannot be obtained.
86	The specified property value is not valid.
87	Standard output and standard error output have timed out.
88	The total standard output and standard error output has exceeded 100 KB.
127	Another error has occurred.

Property list

The properties available for the Terminal Command Plug-in.

Property key	Property name	Description	I/O type	Required
token	Token	Specify the value of the token property of the terminal connect plug-in.	Input	true
commandLine	Command Line	Specify the absolute path of the command or script to be run on the operation target device, using a maximum of 1024 characters. In the command line, specify characters that can be entered in command lines in the OS of the Ops Center Automator server and the OS of the operation target device. Special characters that represent environment variables in the command line are not escaped. To handle a special character as a character string, escape the character with a percent sign	Input	false

Property key	Property name	Description	I/O type	Required
		(%) in Windows, and a backslash (\) in Linux OS. If you must enter the superuser password in the command line to give the user root privileges, specify the reserved.terminal.suPassword reserved property. The reserved.terminal.account, reserved.terminal.password, and reserved.terminal.suPassword reserved properties reference token related authentication information set for the terminal connect plug-in. The specific authentication information the properties reference depends on the setting of the credentialType property of the terminal connect plug-in. ▪ If destination is specified for the credentialType property, the reserved properties reference the authentication information defined in the connection destination. ▪ If property is specified for the credentialType property, the reserved properties reference the authentication information specified in the credentialType property of the terminal connect plug-in.		
charset	Character Set	Specify the character set to use when writing to the standard input of the operation target device and reading from standard output and standard error output. Specify the same character set as that of the user who logs in to the operation target. The names of character sets are not case sensitive. You	Input	false

Property key	Property name	Description	I/O type	Required
		can specify the following character sets: ▪ EUC-JP ▪ eucjp ▪ ibm-943C ▪ ISO-8859-1 ▪ MS932 ▪ PCK ▪ Shift_JIS ▪ UTF-8 ▪ windows-31j		
lineEnd	Newline Character	When Telnet is specified in the protocol property of the terminal connect plug-in, specify the newline character to append to the values specified in the account and password properties. You can specify the following: ▪ CR ▪ LF ▪ CRLF To use 0x0D as the newline character, specify CR. To use 0x0A, specify LF, and to use 0x0D0A, specify CRLF. The default value is CR.	Input	false

Property key	Property name	Description	I/O type	Required
promptPattern	Prompt Pattern	Specify the regular expression pattern to use to detect prompts in standard output and standard error output, using no more than 1,024 characters. This property is used to detect when the operation target device is ready to run commands after the connection is established. Specify the pattern in a PCRE-compliant format. When the output matches the specified regular expression pattern, the plug-in ends immediately. If the output does not match the pattern, the plug-in ends in an error when the time set in the readWaitTime property has elapsed since the last output to standard output or standard error output.	Input	true
readWaitTime	Standard Output Wait Time	When logging in to an operation target device, specify how long to wait after output to standard output or standard error output until the next output. Specify the timeout time in a range from 1 to 86,400,000 (in milliseconds). The default value is 60000	Input	false
outputCondition	Output Condition	Specifies a condition to be output to the standard output property 1-3. You can specify the following values: ▪ always -- Outputs a null character even if it does not match the specified pattern. ▪ patternMatch -- Outputs only when matching the standard output pattern 1-3. If there is no output in output properties, mapped service properties are also not updated.	Input	false

Property key	Property name	Description	I/O type	Required
		The default value is always.		
stdoutProperty1	Standard Output Property 1	The character string extracted by the stdoutPattern1 property is output to this property.	Output	false
stdoutPattern1	Standard Output Pattern 1	Specify the regular expression pattern of the standard output to output to the stdoutProperty1 property, using a maximum of 1,024 characters. Specify the regular expression pattern in a PCRE-compliant format. If you specify the key of a service property in the stdoutProperty1 property but do not specify the stdoutPattern1 property, the entire standard output and standard error output of the command or script specified in the commandLine property is assigned to the service property.	Input	false
stdoutProperty2	Standard Output Property 2	The character string extracted by the stdoutPattern2 property is output to this property.	Output	false
stdoutPattern2	Standard Output Pattern 2	Specify the regular expression pattern of the standard output to output to the stdoutProperty2 property, using a maximum of 1,024 characters. Specify the regular expression pattern in a PCRE-compliant format. If you specify the key of a service property in the stdoutProperty2 property but do not specify the stdoutPattern2 property, the entire standard output and standard error output of the command or script specified in the commandLine property is assigned to the service property.	Input	false

Property key	Property name	Description	I/O type	Required
stdoutProperty3	Standard Output Property 3	The character string extracted by the stdoutPattern3 property is output to this property.	Output	false
stdoutPattern3	Standard Output Pattern 3	Specify the regular expression pattern of the standard output to output to the stdoutProperty3 property, using a maximum of 1,024 characters. Specify the regular expression pattern in a PCRE-compliant format. If you specify the key of a service property in the stdoutProperty3 property but do not specify the stdoutPattern3 property, the entire standard output and standard error output of the command or script specified in the commandLine property is assigned to the service property.	Input	false
defaultReturnCode	Default Return Code	Specify the value to return as the return code when standard output and standard error output do not match the regular expression pattern specified in the returnCodePattern property. Specify a value in the range from 0 to 63. The default is 0.	Input	false
returnCodePattern	Return Code Pattern	Specify the regular expression pattern for standard output and standard error output, using a maximum of 1,024 characters. Specify the pattern in a PCRE compliant format. If standard output and standard error output match the specified pattern, the plug-in returns the value specified in the returnCode property.	Input	false

Property key	Property name	Description	I/O type	Required
returnCode	Return Code	Specify the return code to be returned by the plug-in when standard output and standard error output match the pattern set in the returnCodePattern property. You can specify a value in the range from 0 to 63. If you omit this property, the plug-in returns the value specified in the defaultReturnCode property.	Input	false

Usage examples of terminal command plug-in

Example of terminating a terminal command plug-in with an error when an error is output to standard output

The following table describes an example of a terminal command plug-in that ends with an error when it acquires error-related information from standard output. Set the plug-in property as follows:

Property key	Example of specified value	Meaning of specified value
commandLine	configServer arg0 arg1 arg2	Runs the specified command or script.
promptPattern	^\[prompt\]	If the contents of standard output matches [prompt], the plug-in ends and determines the return value.
stdoutPattern1	^Message:(.*)	The character string following Message: in standard output is stored in the stdoutProperty1 property.
stdoutPattern2	^Error:(.*)	The character string following Error: in standard output is stored in the stdoutProperty2 property.
stdoutPattern3	^ReturnCode:(.*)	The character string following Returncode: in standard output is stored in the stdoutProperty3 property.
defaultReturnCode	0	If the contents of standard output do not match the value specified in the returnCodePattern property, return code 0 is returned.

Property key	Example of specified value	Meaning of specified value
returnCodePattern	^Error:	If the contents of standard output match Error:, the plug-in returns the return code specified in the returnCode property.
returnCode	1	If the contents of standard output matches the value specified in the returnCodePattern property, the plug-in returns code 1.

The following describes the function of a plug-in with the previously listed properties when it encounters the following standard output.

```
configServer arg0 arg1 arg2
Message:command failed
Error:Permission Denied
ReturnCode:128
[prompt]>
```

The contents of standard output match the value specified in the promptPattern property, so the terminal command plug-in determines which return code to return. Because standard output matches the value specified in the returnCodePattern property, the plug-in returns code (1), the value specified in the returnCode property.

The character strings extracted by the properties stdoutPattern1 to stdoutPattern3 are stored as follows in the properties stdoutPrpoerty1 to stdoutProperty3:

- stdoutProperty1: command failed
- stdoutProperty2: Permission Denied
- stdoutProperty3: 128

Example of sending a GET request to an HTTP server

The following describes how to configure a plug-in that issues a request such as the following one to an HTTP server and verifies the response.

```
GET /index.html HTTP/1.1
Host: ServerA
User-Agent: Automation Director
Accept-Charset: UTF-8
```

To issue a GET request to an HTTP server, specify each line of the request method and request header in the commandLine property of a terminal command plug-in.

Because the last line of the request needs to be blank, you must run the terminal command plug-in five times. The following table describes examples of the values to set in the properties of each instance of the plug-in.

Order of execution	Value specified in the commandLine	Value specified in the lineEnd	Value specified in promptPattern
First	GET /index.html HTTP/1.1	CRLF	.*
Second	Host: ServerA	CRLF	.*
Third	User-Agent: Ops Center Automator	CRLF	.*
Fourth	Accept-Charset: UTF-8	CRLF	.*
Fifth	-- (adds a blank line. Do not specify a value)	CRLF	.*

Because HTTP server requests use [CR]+[LF] as delimit characters, specify CRLF for the value specified in the lineEnd.

In the promptPattern property of the first to fourth terminal command plug-ins, you can specify regular expression patterns that also match blank characters.

Because standard output continues after you run the terminal command plug-ins, specify a regular expression that detects the end of standard output by the terminal command plug-ins in the promptPattern property.

The following describes the function of a plug-in with the previously listed properties when it encounters the following standard output:

```
HTTP/1.1 200 OK
Date: Mon, 18 Mar 2013 10:19:20 GMT
Server: Cosminexus HTTP Server
Last-Modified: Sun, 31 Jul 2005 05:27:52 GMT
ETag: "2d000000012d48-f-3fd2b60590600"
Accept-Ranges: bytes
Content-Length: 15
Content-Type: text/html

<HTML></HTML>
```

Because the contents of standard output match the value specified in the promptPattern property, the terminal command plug-in determines the return code.

If standard output matches the value specified in the returnCodePattern property, the return code specified in the returnCode property is returned as the return code of the plug-in.

If standard output does not match the value specified in the returnCodePattern property, the plug-in returns the return code specified in the defaultReturnCode property.

Terminal Disconnect Plug-in

The Terminal Disconnect Plug-in ends a connection established with an operation target device by a terminal connect plug-in.

Prerequisite

- Terminal disconnect plug-in uses the protocol specified in the protocol property of the terminal connect plug-in to communicate with the Ops Center Automator server.

Cautionary notes

- If a task is stopped while a plug-in is running, the status of the task changes to Failed or Completed when the processing of the terminal disconnect plug-in finishes. The status of steps and tasks after plug-in finishes depends on the return code of the step and the condition for running subsequent steps. You can set Subsequent-step Execution Condition in the **Create Step** dialog box or the **Edit Step** dialog box.
- If you forcibly end a task while a plug-in is running, reading from standard output and prompt detection are canceled and the task enters Failed status. The session and token are then discarded. In this case, the return code of the step in the Task Details dialog box is -1. The return code output to the task log depends on the timing of when the task was forcibly ended.

Return codes

The Terminal Disconnect Plug-in generates the following return codes:

Return Code	Description
0	The plug-in ended normally. The plug-in ends normally even if the connection is already closed.
65	The connection with the Ops Center Automator server failed. For example, the Ops Center Automator server might have stopped while the plug-in was running.
66	The following user is mapped to the Ops Center Automator user: ▪ A user who does not belong to the Administrators group. ▪ A user other than the built-in Administrator who belongs to the Administrators group, in an environment with UAC enabled.
68	No information about the target job execution ID exists.
69	An environment variable of the task-processing engine cannot be acquired.
80	Task execution has stopped.
81	The plug-in was called in a status that is not valid.

Return Code	Description
82	The request message from the task-processing engine cannot be correctly parsed.
83	The environment of the Ops Center Automator server is corrupted.
84	Information about the specified plug-in cannot be obtained.
86	The specified property value is not valid.
127	Another error has occurred.

Property list

The following properties are available for the Terminal Disconnect Plug-in:

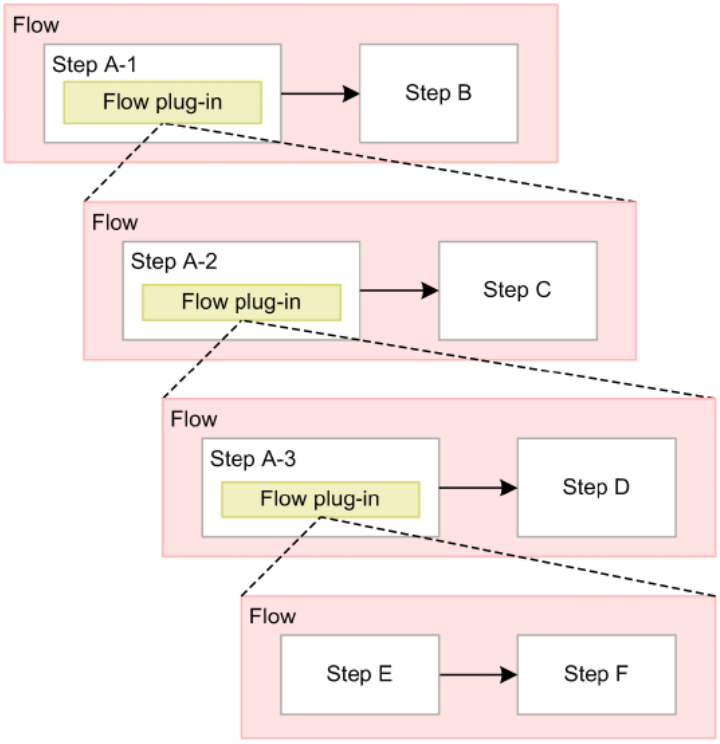
Property key	Property name	Description	I/O type	Required
token	Token	Specify the value of the token property of the terminal connect plug-in.	Input	true

Flow Plug-in

You can define a flow hierarchy using the Flow Plug-in.

The Flow Plug-in allows you to create hierarchical flows by defining flows within other flows. You can define a maximum of 25 hierarchical levels, with the top-level flow being level 1.

The following figure shows how the flow is established.



Cautionary notes

- If execution of a task is stopped while a plug-in is running, the status of the task changes to Failed or Completed when the step running in the flow plug-in ends.
- The return code of a flow plug-in is always 0. If a step within a hierarchy flow ends abnormally, the plug-in returns code 0. The return code of the flow plug-in does not reflect the return codes of the constituent steps of the hierarchy flow.

Return codes

The Flow Plug-in generates the following return codes:

- 0: The plug-in ended normally.
- 1: A step in a lower execution flow ended with a warning.
- 2: A step in a lower execution flow ended abnormally.

Note: If the step execution of a lower flow fails, the task does not end at that point. Determination of whether the task continues or stops is made by setting the Next Step Condition of the flow plug-in step.

Property list

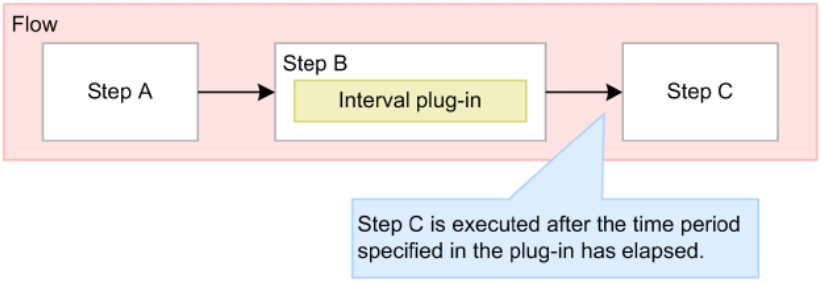
The following properties are available for the Flow Plug-in:

Property key	Property name	Description	I/O type
errorStep	Step in which error occurred	The step ID that failed in one lower flow is output in half-width comma-separated.	Output
returnValueOfErrorStep	Return value of step in which error occurred	The return value of the step that failed in one lower flow is output in half-width comma separated values.	Output

Interval Plug-in

The Interval Plug-in controls the interval between running steps. The user specifies the wait time for a process as the execution interval, and Ops Center Automator waits for the interval to elapse before running the succeeding steps. By using an interval plug-in, you can run steps at fixed intervals.

The following figure shows how the Interval Plug-in is used.



Cautionary notes

- The communication status and other factors might cause a discrepancy between the actual wait time and the time specified by the plug-in.
- You cannot change the property values when you run the service. Set the values when you create the flow.
- You can only specify literal characters in the input property. You cannot map the value of a service property or reserved property.
- If execution of a task is stopped during plug-in execution, the task enters Failed or Completed status after the interval plug-in has finished processing.

Return codes

The Interval Plug-in generates the following return codes:

Return Code	Description
0	The plug-in ended normally. The plug-in ends normally even if the connection has already been closed.
18	A task was forcibly ended during plug-in execution.
1 to 17, 19 or higher	The plug-in ended abnormally. Use the **`hcmds64getlogs`** command to acquire log information and identify the problem.

Property list

The following properties are available for the Interval Plug-in:

Property key	Property name	Description	I/O type
interval [R]	Interval	This property specifies how long to wait before running the next step, in the range from 1 to 1,440 (minutes). The default value is ten minutes.	Input
[R]: Required			

Branch by ReturnCode Plug-in

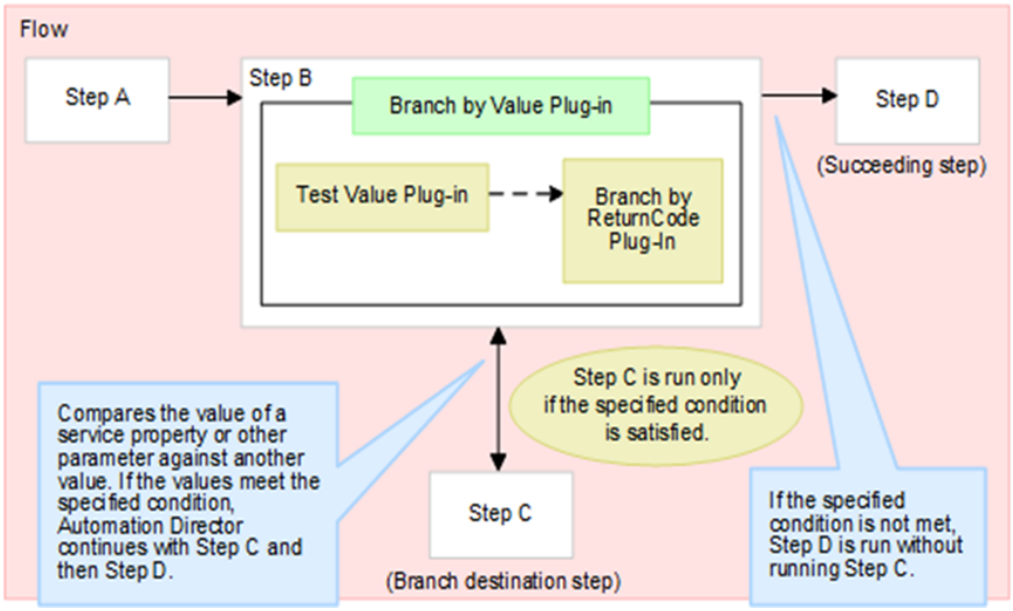
The Branch by ReturnCode Plug-in branches the flow of processing based on the return value of the previous step.

This plug-in allows you to select which step to run next based on the return code of the preceding step.

A Branch by ReturnCode Plug-in connects to two branch destination steps: A succeeding step, and a step that is only run when specific conditions are met. If the return code matches the specified condition, Ops Center Automator runs the branch destination step and the succeeding step, in that order. If the return code does not match the specified conditions, Ops Center Automator runs the succeeding step only.

By using this plug-in together with a test value plug-in, you can select the steps in a flow based on a character string.

The following figure shows how the Branch by ReturnCode Plug-in is used.



Cautionary notes

- When a task is stopped or forcibly ended during plug-in execution, the task enters Completed status after the Branch by ReturnCode Plug-in finishes processing.
- You cannot change the property values when you run the service. Set the values when you create the flow.
- You can only specify literal characters in input properties. You cannot map the value of a service property or reserved property.
- If the Branch by ReturnCode Plug-in stops processing, use the **`hcmds64getlogs`** command to acquire log information and identify the problem.

Return codes

The Branch by ReturnCode Plug-in generates the following return codes:

- 0 or higher: The plug-in ended normally. The return code of the preceding step of the Branch by ReturnCode Plug-in is set as the return code.

Property list

The following properties are available for the Branch by ReturnCode Plug-in:

Property key	Property name	Description	I/O type
condition R	Condition	Specify the condition for the return code of the preceding step. You can choose from the following conditions: ▪ ReturnCode=value1 The return code is equal to Value1. ▪ ReturnCode!=value1 The return code is not equal to Value1. ▪ ReturnCode<value1 The return code is less than Value1. ▪ ReturnCode>value1 The return code is greater than Value1. ▪ ReturnCode<=value1 The return code is less than or equal to Value1. ▪ ReturnCode>=value1 The return code is greater than or equal to Value1. ▪ ReturnCode>value1 AND ReturnCode<value2 The return code is greater than Value1 and less than Value2. ▪ ReturnCode>=value1 AND ReturnCode<value2 The return code is greater than or equal to Value1, and less than Value2. ▪ ReturnCode>value1 AND ReturnCode<=value2 The return code is greater than Value1, and less than or equal to Value2. ▪ ReturnCode>=value1 AND ReturnCode<=value2 The return code is greater than or equal to Value1, and less than or equal to Value2.	Input

Property key	Property name	Description	I/O type
		▪ ReturnCode<value1 OR ReturnCode>value2 The return code is less than Value1, or greater than Value2. ▪ ReturnCode<=value1 OR ReturnCode>value2 The return code is less than or equal to Value1, or greater than Value2. ▪ ReturnCode<value1 OR ReturnCode>=value2 The return code is less than Value1, or greater than or equal to Value2. ▪ ReturnCode<=value1 OR ReturnCode>=value2 The return code is less than or equal to Value1, or greater than or equal to Value2. The default value is ReturnCode=value1.	
value1 [R]	Value1	Specify a numerical value against which to judge the return code, within the range from 0 to 999. The value is mapped to value1 in the condition property. The default value is 0.	Input
value2	Value2	Specify a numerical value against which to judge the return code, within the range from 0 to 999. The value is mapped to value2 in the condition property. This value takes effect when value2 is included in a condition property. The default value is 0.	Input
[R]: Required			

Example of a property specification

A Branch by ReturnCode Plug-in determines whether the return code is within a specified range of values.

The following describes the range of valid condition values, using the following values of the condition, value1, and value2 properties as examples.

A. The return code is 25 or greater and less than 75

condition (Condition): ReturnCode>=value1 AND ReturnCode<value2

value1 (Value1): 25

value2 (Value2): 75

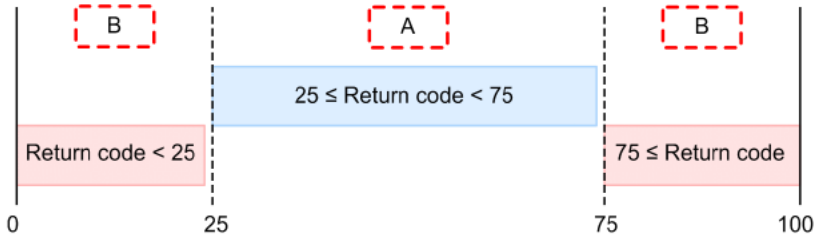
B. The return code is less than 25, or 75 or higher

condition (Condition): ReturnCode<value1 OR ReturnCode>=value2

value1 (Value1): 25

value2 (Value2): 75

The following figure shows the range of return codes that match each condition.



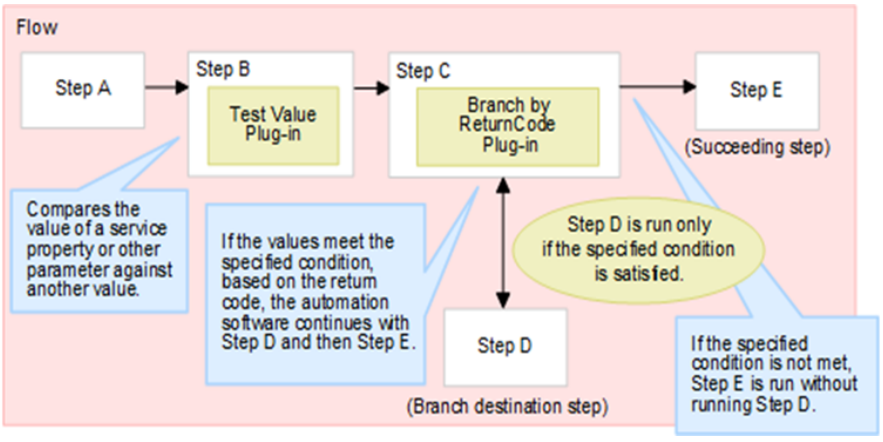
Test Value Plug-in

The Test Value Plug-in compares service property values and returns 0 if the values match the conditions.

The plug-in compares the value of a service property, the value of a reserved property, a literal string, or any combination of these values against a specified value. If the condition is met, the plug-in returns 0.

By using this plug-in together with a Branch by ReturnCode Plug-in, you can select the steps in a flow based on a character string.

The following figure shows how the Test Value Plug-in is used.



Cautionary notes

- If a task is stopped while a plug-in is running, the task enters Completed status after the Test Value Plug-in finishes processing.

Return codes

The Test Value Plug-in generates the following return codes:

Return Code	Description
0	The value matched the condition. Alternatively, 0 is specified in the defaultReturnCode property.
1	The value did not match the condition. Alternatively, 1 is specified in the defaultReturnCode property.
63	Condition failed. 63 is specified in the defaultReturnCode property.
65	The connection with the Ops Center Automator server failed. For example, the Ops Center Automator server might have stopped while the plug-in was running.
66	The following user is mapped to the Ops Center Automator user: ▪ A user who does not belong to the Administrators group. ▪ A user other than the built-in Administrator who belongs to the Administrators group, in an environment with UAC enabled.
68	No information about the target job execution ID exists.
69	An environment variable of the task-processing engine cannot be acquired.
80	Task execution has stopped.
81	The plug-in was called in a status that is not valid.
82	The request message from the task-processing engine cannot be correctly parsed.
83	The environment of the Ops Center Automator server is corrupted.
84	Information about the specified plug-in cannot be obtained.
86	The specified property value is not valid.
127	Another error has occurred.

Property list

The following properties are available for the Test Value Plug-in:

Property key	Property name	Description	I/O type
condition [R]	Condition	Specify the judgment condition for the valueX property. You can select from the following conditions: ▪ valueX=value1 ValueX and Value1 are equal (numerical comparison). ▪ valueX!=value1 ValueX and Value1 are not equal (numerical comparison). ▪ valueX<value1 ValueX is less than Value1 (numerical comparison). ▪ valueX>value1 ValueX is greater than Value1 (numerical comparison). ▪ valueX<=value1 ValueX is less than or equal to Value1 (numerical comparison). ▪ valueX>=value1 ValueX is greater than or equal to Value1 (numerical comparison). ▪ valueX>value1 AND valueX<value2 ValueX is greater than Value1 and less than Value2 (numerical comparison) ▪ valueX>=value1 AND valueX<value2 ValueX is greater than or equal to Value1, and less than Value2 (numerical comparison). ▪ valueX>value1 AND valueX<=value2 ValueX is greater than Value1, and less than or equal to Value2 (numerical comparison).	Input

Property key	Property name	Description	I/O type
		▪ valueX>=value1 AND valueX<=value2 ValueX is greater than or equal to Value1, and less than or equal to Value2 (numerical comparison). ▪ valueX<value1 OR valueX>value2 ValueX is less than Value1, or greater than Value2(numerical comparison). ▪ valueX<=value1 OR valueX>value2 ValueX is less than or equal to Value1, or greater than Value2 (numerical comparison). ▪ valueX<value1 OR valueX>=value2 ValueX is less than Value1, or greater than or equal to Value2 (numerical comparison). ▪ valueX<=value1 OR valueX>=value2 ValueX is less than or equal to Value1, or greater than or equal to Value2 (numerical comparison). ▪ valueX equals value1 ValueX and Value1 are equal. Values are case sensitive (character string comparison). ▪ valueX not equals value1 ValueX and Value1 are not equal. Values are case sensitive (character string comparison). ▪ valueX contains value1 ValueX contains Value1. Values are case sensitive (character string comparison). ▪ valueX not contains value1 ValueX does not contain Value1. Values are case sensitive (character string comparison).	

Property key	Property name	Description	I/O type
		The default value is valueX=value1.	
valueX [R]	ValueX	Specify a value as the basis for comparison, using no more than 1,024 characters. You can use the following formats individually or combined. ▪ ?dna_service-property-key? (when referencing the value of a service property) ▪ ?dna_reserved-property-key? (when referencing the value of a reserved property) ▪ literal-string	Input
value1	Value1	Specify the value against which to compare the valueX property, using no more than 1,024 characters. You can use the following formats individually or together. ▪ ?dna_service-property-key? (when referencing the value of a service property) ▪ ?dna_reserved-property-key? (when referencing the value of a reserved property) ▪ literal-string The value is mapped to value1 in the condition property.	Input
value2	Value2	Specify the value against which to compare the valueX property, using no more than 1,024 characters. You can use the following formats individually or together. ▪ ?dna_service-property-key? (when referencing the value of a service property) ▪ ?dna_reserved-property-key? (when referencing the value of a reserved property) ▪ literal-string	Input

Property key	Property name	Description	I/O type
		The value is mapped to value2 in the condition property. The value in this property takes effect when value2 is specified in the condition property.	
defaultReturnCode [R]	Default Return Code On Error	This property specifies the value returned by the plug-in when a numerical comparison is specified in the condition property, and a value that cannot be compared on a numerical basis is specified in any of the valueX, value1, and value2 properties. ▪ **0** Specify 0 when using "The value matched the judgment condition." as the judgment result. ▪ **1** Specify 1 when using "The value did not match the judgment condition." as the judgment result. ▪ **63** Specify 63 when using "Judgment failed" as the judgment result to make the step end abnormally. The default value is 63.	Input
[R]: Required			

Example of property specification

A test value plug-in determines whether an input value is within a specified range of values.

The following describes the range of valid condition values, using the following values of the condition, value1, and value2 properties as examples.

A. The input value is greater than or equal to 25 and less than 75

condition (Condition): ReturnCode>=value1 AND ReturnCode<value2

value1 (Value1): 25

value2 (Value2): 75

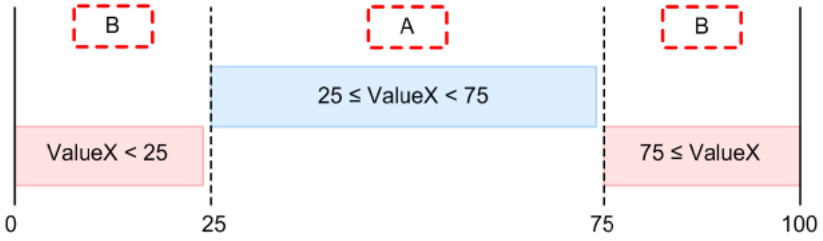
B. The input value is less than 25, or 75 or greater

condition (Condition): ReturnCode<value1 OR ReturnCode>=value2

value1 (Value1): 25

value2 (Value2): 75

The following figure shows the range of return codes that match each condition.



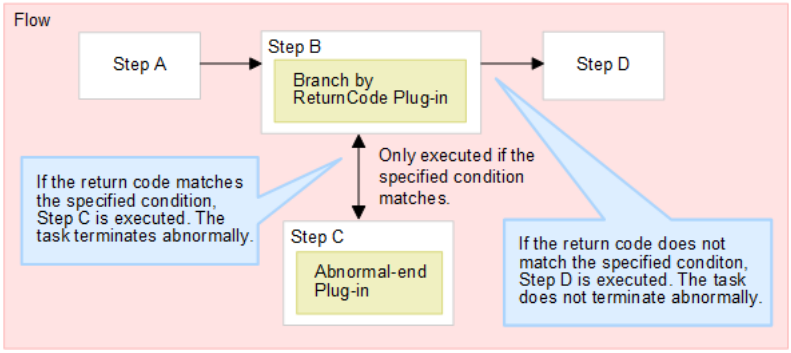
Abnormal-End Plug-in

The Abnormal-End Plug-in handles the abnormal termination of a running flow, task, hierarchical flow, or repeatedly run flow.

The plug-in lets you end a running task abnormally.

By using this plug-in together with a Branch by ReturnCode Plug-in, you can also end a flow abnormally when a judgment condition is met.

The following figure shows how the Abnormal-End Plug-in is used.



Cautionary notes

- If a task is stopped while running a plug-in, the task enters abnormal termination status after the abnormal-end plug-in finishes processing.
- If you use an abnormal-end plug-in within a flow plug-in, the hierarchical flow and any higher-level flows that feature flow plug-ins also end abnormally. Running tasks also end abnormally, and the flow plug-in returns 0.
- When you use an abnormal-end plug-in in the context of a repeated execution plug-in, the repeated execution plug-in returns 1 if the repeated processing ends abnormally one time. If every instance of the repeated processing ends abnormally, the repeated execution plug-in returns 2.

Return codes

The Abnormal-End Plug-in generates the following return codes:

- 0: The plug-in ended normally (the step was ended abnormally).
- 80: Task execution has stopped.

Branch by Property Value Plug-in

The Branch by Property Value Plug-in branches the flow of processing based on service property values.

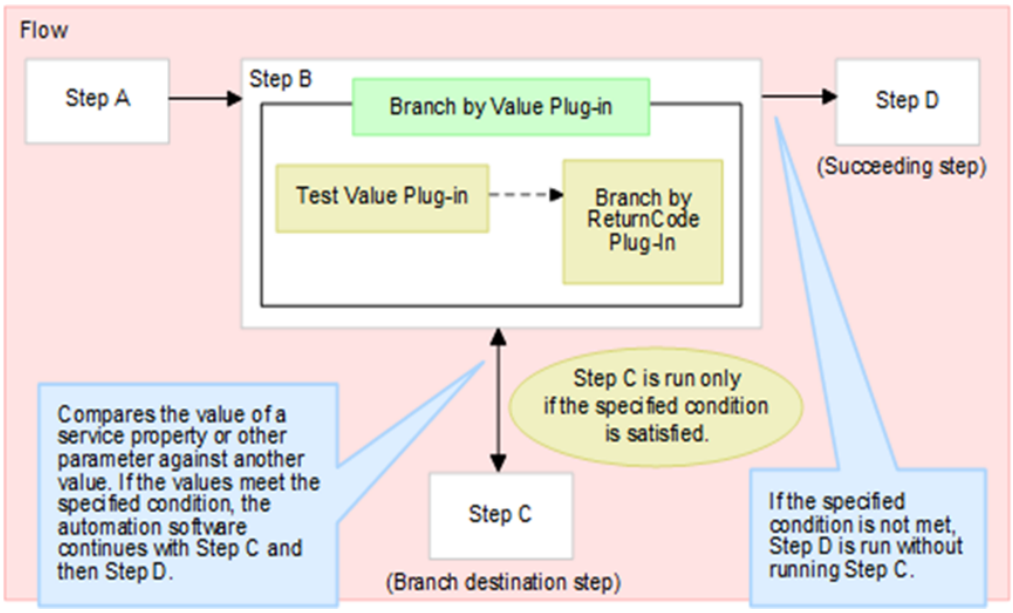
The plug-in compares the value of a service property, the value of a reserved property, a literal string, or any combination of these values against a specified value. The result of the comparison determines which step Ops Center Automator runs next.

This Branch by Property Value Plug-in allows you to select which step to run next based on the return code of the preceding step.

This plug-in connects to two branch destination steps: A succeeding step, and a step that is only run when the judgment condition is met. If the input value matches the condition, Ops Center Automator runs the branch destination step and the succeeding step, in that order. If the input value does not match the conditions, Ops Center Automator runs the succeeding step only.

This plug-in combines the functionality of a test value plug-in and a Branch by Returncode plug-in.

The following figure shows how the Branch by Property Value Plug-in is used.



Cautionary notes

- When you run this plug-in, the information output to the task log reflects the execution of the test value plug-in. The Branch by Property Value Plug-in does not contribute to the task log.
- If a task is stopped while the plug-in is running, the task enters Completed status after the Branch by Property Value Plug-in finishes processing.

Return codes

The Branch by Property Value Plug-in generates the following return codes:

Return Code	Description
0	The plug-in returns 0 when: ▪ The condition result is true. ▪ A numerical comparison is specified in the condition property, a value that cannot be compared on a numerical basis is specified in any of the valueX, value1, and value2 properties, and 0 is specified in the defaultReturnCode property.
1	The plug-in returns 1 when: ▪ The condition result is false. ▪ A numerical comparison is specified in the condition property, a value that cannot be compared on a numerical basis is specified in any of the valueX, value1, and value2 properties, and 1 is specified in the defaultReturnCode property.
80	The plug-in returns 80 when task execution is stopped.

Property list

The following properties are available for the Branch by Property Value Plug-in:

Property key	Property name	Description	I/O type
condition [R]	Condition	Specify the branch condition for the valueX property. You can select from the following conditions: ▪ valueX>=value1 ValueX is greater than or equal to Value1 (numerical comparison). ▪ valueX>value1 AND valueX<value2 ValueX is greater than Value1 and less than Value2 (numerical comparison).	Input

Property key	Property name	Description	I/O type
		▪ valueX>=value1 AND valueX<value2 ValueX is greater than or equal to Value1, and less than Value2 (numerical comparison). ▪ valueX>value1 AND valueX<=value2 ValueX is greater than Value1, and less than or equal to Value2 (numerical comparison). ▪ valueX>=value1 AND valueX<=value2 ValueX is greater than or equal to Value1, and less than or equal to Value2 (numerical comparison). ▪ valueX<value1 OR valueX>value2 ValueX is less than Value1, or greater than Value2 (numerical comparison). ▪ valueX<=value1 OR valueX>value2 ValueX is less than or equal to Value1, or greater than Value2 (numerical comparison). ▪ valueX<value1 OR valueX>=value2 ValueX is less than Value1, or greater than or equal to Value2 (numerical comparison). ▪ valueX<=value1 OR valueX>=value2 ValueX is less than or equal to Value1, or greater than or equal to Value2 (numerical comparison). ▪ valueX equals value1 ValueX and Value1 are equal. This judgment is case sensitive (character string comparison). ▪ valueX not equals value1 ValueX and Value1 are not equal. This judgment is case sensitive (character string comparison).	

Property key	Property name	Description	I/O type
		▪ valueX contains value1 ValueX contains Value1. This judgment is case sensitive (character string comparison). ▪ valueX not contains value1 ValueX does not contain Value1. This judgment is case sensitive (character string comparison). The default value is valueX=value1.	
valueX [R]	ValueX	Specify a value as the basis for comparison, using no more than 1,024 characters. You can use the following formats individually or combined. ▪ ?dna_service-property-key? (when referencing the value of a service property) ▪ ?dna_reserved-property-key? (when referencing the value of a reserved property) ▪ literal-string	Input
value1	Value1	Specify the value against which to compare the valueX property, using no more than 1,024 characters. You can use the following formats individually or together. ▪ ?dna_service-property-key? (when referencing the value of a service property) ▪ ?dna_reserved-property-key? (when referencing the value of a reserved property) ▪ literal-string The value is mapped to value1 in the condition property. The value in this property takes effect when value1 is specified in the condition property.	Input
value2	Value2	Specify the value against which to compare the valueX property, using no more than 1,024 characters. You can use	Input

Property key	Property name	Description	I/O type
		the following formats individually or together. ▪ ?dna_service-property-key? (when referencing the value of a service property) ▪ ?dna_reserved-property-key? (when referencing the value of a reserved property) ▪ literal-string The value is mapped to value2 in the condition property. The value in this property takes effect when value2 is specified in the condition property.	
defaultReturnCode [R]	Default Return Code On Error	This property specifies the value returned by the plug-in when a numerical comparison is specified in the condition property, and a value that cannot be compared on a numerical basis is specified in any of the valueX, value1, and value2 properties. ▪ **0** Specify 0 when using "The value matched the judgment condition." as the judgment result. ▪ **1** Specify 1 when using "The value did not match the judgment condition." as the judgment result. ▪ **63** Specify 63 for the step to end abnormally when the judgment gives a failed result. The plug-in ends abnormally without executing the branch destination step or the succeeding step. The default value is 63.	Input
[R]: Required			

File Export Plug-in

Exports the input content to a file.

The File Export Plug-in can output input values of any format to a file. You can use the Apache Velocity Engine VTL (Velocity Template Language) http://velocity.apache.org/ to specify the output format. If no format is specified, the input values are output as-is with no formatting.

The File export plug-in specifies the full-path to the output file, exports the data, and generates error messages if necessary

Prerequisites

- Name and location of the file to export.
- If necessary, prepared template for output format.

Cautionary notes

- If a file has the same name as an existing output file, the former will be overwritten.
- If a folder has the same name as an existing output file, a write error occurs.
- Line feed codes in the export content and output template are included in the output file.

Return codes

The File Export Plug-in generates the following return codes.

Return Code	Description
0	Ended normally.
1	The length of the output file path is longer than 256 characters.
2	A grammatical error has been detected in the VTL description.
3	This value is returned when an undefined attribute or method is detected in the VTL description. Usually, this value is not returned unless Velocity is run in strict mode. By default, strict mode is set to false.
4	The resource file used for Velocity cannot be located. In most cases, this value is not returned since the File Export Plug-in does not use the resource file but is included to accommodate the exception that is generated by the Velocity API.
5	An error occurred while writing the output file.
63	An error occurred during the processing of the plug-in.
80	Task execution has stopped.

Property list

The following properties are available for the File Export Plug-in:

Property key	Property name	Description	I/O type
content [R]	Content To Export	Specifies the content to be exported to the file.	Input
fileName [R]	Output File Name	Specifies the name of the file where the content is to be exported. Empty strings are not valid. The length of the full path consisting of the output folder path and the output file name cannot exceed 256 characters.	Input
directoryPath	Output Directory Path	Folder path where the output file is to be created; an absolute or relative path can be specified. If you specify a relative path, the task working folder (Automation/data/task/[tasked]) is assumed as the starting point for the path. If you do not specify a folder path, the task working folder (Automation/data/task/[tasked]) is used. The length of the full path consisting of the output folder path and the output file name cannot exceed 256 characters.	Input
template	Template For The Output	Specifies a template with the VTL description that defines the format for the output content.	Input
charset	Character Set	Specifies the character set for the output file.	Input
exportFilePath	Output File Path	Specifies the full path for the output file to be exported.	Output
message	Message	Specifies the log for messages generated during the processing of the plug-in.	Output
[R]: Required			

Input values with the output template

Any character can be specified in the input values. If the input values are JSON, the values are JSON decoded as objects via the variable $root in the VTL description of the output template, and the objects can be viewed. If the input values are not JSON, $root keeps the input values as character strings. The description method of the output template complies with the VTL syntax. Note that the variable $root described previously is a reserved variable to which File Export Plug-in specifies values in advance. In addition, File Export Plug-in has a utility for CSV output as a reserved variable $csv. The following table shows the reserved variables that can be used in the VTL description for File Export Plug-in.

Variable name	Description
$root	If an input value is JSON, the value is JSON decoded and stored as an object. If the input value is not JSON, the value is stored as a character string.
$csv Gives the following utility methods for CSV output. These can be used for the VTL description using Velocity's ToolManager.	$csv.value(String) -- Takes a single string as an argument; if a character (such as a double quotation mark, comma, or line feed) that needs an escape in CSV is included, enclose the value with double quotation marks. If a double quotation mark is included in a string value, the string in which the double quotation mark is escaped (" --> "") is output. (Output process as a single cell of the CSV format)
	$csv.values(String…) -- Takes multiple strings as arguments (variable-length arguments) and outputs the strings into a single line with the CSV format; the output format of a single cell is same as the value(String) function described previously.
	$csv.values(Collection<String>) -- Takes an array and outputs the array into a single line with the CSV format; the output format of a single cell is the same as the value(String) function described previously.

JavaScript Plug-in

The JavaScript Plug-in Runs a specific JavaScript code. The plug-in can run any specified JavaScript code as follows:

- To view any service property and plug-in property value.
- To pass the maximum of 10 arguments via plug-in properties to the JavaScript code.
- To keep the maximum of 10 values in plug-in output properties.
- To return a value and keep the return value in the output properties.

Return codes

The JavaScript Plug-in generates the following return codes:

Return Code	Description
0	Ended normally.
1	Script has ended due to an error.
60	A JavaScript library read error has occurred.
61	A JavaScript compile error has occurred.
62	The JavaScript code is not properly formatted.
63	An error occurred during the processing of the plug-in.
80	Task execution has stopped.

Property list

The following properties are available for the JavaScript Plug-in:

Property key	Property name	Description	Default value	I/O type	Required
scriptBody	Script body	Specifies the JavaScript code strings.	--	Input	true
importedScript	Imported script	Specifies methods and constants (code string of JavaScript) to be used in common with other JavaScript Plug-ins placed on the same service template. The following items can be used in the script of importedScript: ▪ JS library: Same as the JS library available at scriptBody. ▪ Functions: print() function. Same as the print() function available at scriptBody.	-	Input	false
arg0	Argument(0)	Specifies an argument to be passed to the script.	--	Input	false
arg1	Argument(1)	Specifies an argument to be passed to the script.	--	Input	false

Property key	Property name	Description	Default value	I/O type	Required
arg2	Argument(2)	Specifies an argument to be passed to the script.	--	Input	false
arg3	Argument(3)	Specifies an argument to be passed to the script.	--	Input	false
arg4	Argument(4)	Specifies an argument to be passed to the script.	--	Input	false
arg5	Argument(5)	Specifies an argument to be passed to the script.	--	Input	false
arg6	Argument(6)	Specifies an argument to be passed to the script.	--	Input	false
arg7	Argument(7)	Specifies an argument to be passed to the script.	--	Input	false
arg8	Argument(8)	Specifies an argument to be passed to the script.	--	Input	false
arg9	Argument(9)	Specifies an argument to be passed to the script.	--	Input	false
notify	Notification flag	Specifies a non-empty string if the script detects something to notify; the plug-in ends with a return value of 1 if a non-empty string is specified here.	--	Output	false
returnValue	Return value	The content of the object returned from the function of the specified script is output as a character string.	--	Output	false
out0	Output(0)	A value set to the out0 key in the map of the second argument in the user-specified script is output.	--	Output	false
out1	Output(1)	A value set to the out1 key in the map of the second argument in the user-specified script is output.	--	Output	false

Property key	Property name	Description	Default value	I/O type	Required
out2	Output(2)	A value set to the out2 key in the map of the second argument in the user-specified script is output.	--	Output	false
out3	Output(3)	A value set to the out3 key in the map of the second argument in the user-specified script is output.	--	Output	false
out4	Output(4)	A value set to the out4 key in the map of the second argument in the user-specified script is output.	--	Output	false
out5	Output(5)	A value set to the out5 key in the map of the second argument in the user-specified script is output.	--	Output	false
out6	Output(6)	A value set to the out6 key in the map of the second argument in the user-specified script is output.	--	Output	false
out7	Output(7)	A value set to the out7 key in the map of the second argument in the user-specified script is output.	--	Output	false
out8	Output(8)	A value set to the out8 key in the map of the second argument in the user-specified script is output.	--	Output	false
out9	Output(9)	A value set to the out9 key in the map of the second argument in the user-specified script is output.	--	Output	false

You specify the plug-in input/output properties in the property list. Combinations of service property values, reserved property values, and literal characters can be used for the input properties.

JavaScript code specifiable for the JavaScript body

The following table shows the JavaScript codes that are allowable in the body of the script:

Character encoding			UTF-8	
Available JS Library			underscore.js 1.8.3 auto_util-1.0.1.js (Bundled library of Ops Center Automator)	
Format			Must be an unnamed function (See the sample code.)	
Function call interface	Argument	serviceProperties	Object type	Service input properties are mapped. The values of the service properties can be viewed from the script. Note that even if the script updates, deletes, or adds any of the values, the modified values will not be reflected in the service properties after the script is called.
		pluginProperties	Object type	Plug-in properties are mapped. The values of the plug-in properties can be viewed from the script.

				arg0 to arg9	Strings specified in the plug-in properties are directly mapped. Even the strings conform to JSON, if the script refers to here, and arguments can be obtained as strings, not objects. When you do not expect the validation JSON strings to objects described afterward, verify here.
				notify	If you specify a value other than a non-empty string to a member specified for notify in the script, the plug-in finishes with a return value of 1 after calling.

				out0 to out9	If you specify values to members specified for out0 to out9 in the script, the values are reflected in the plug-in output properties of Argument(0) to Argument(9) after calling. The values are reflected in the processing results of the script, and the results will be used for the next step. (See the sample code.)
		arg0 to arg9	Plug-in properties Argument(0) to Argument(9) are mapped. (Optional) If you specify JSON strings to the plug-in properties, the strings can be obtained as objects in the functions of the script. ▪ Strings that are enclosed with double quotation marks are mapped as is. ▪ If strings are not enclosed, strings that fail JSON validation are mapped as is.		
	Return code	Any objects can be returned from the script. After calling the script, the return codes are extracted as JSON strings that are then reflected in the plug-in output property Return value. The values are reflected in the processing results of the script and then used for the next step. (See the sample code.)			
print () function	By using the print() function in the script, you can output any strings to the task log. In this case, choose a log level by adding a specific prefix to the beginning of the string. Note that alphabetical prefixes are case-sensitive. (See the sample code.)				

	Prefix	[Severe]	Outputs as log level 0
		[Information]	Outputs as log level 10
		[Fine]	Outputs as log level 20
		[Finer]	Outputs as log level 30
		[Debug]	Outputs as log level 40
		(No prefix)	Same as the prefix [Information]
Other functionalities	If an exception is thrown in the script or an unexpected exception occurs in the script, the plug-in ends abnormally and shows the exception in the task log.		
	When the script finishes properly, and when the values returned from the script or the values specified in the plug-in output properties (out0 to out9) of the script contain null or undefined, those values are stored as null or undefined.		

auto util Library

The JavaScript plug-in supports the auto util library with the following features:

- Sends an http or https request to any destination.
- The setting values of Web Service Connection can be referred to as the connection information.

The following table shows the auto util methods that are available:

Table 12 auto util methods

Method Name	Argument	Return Value	Description
sleep	Specify the time to sleep in a numeric type (in milliseconds)	None	Sleep for the specified time.
parseJson	Specify a string representation of JSON by string type.	JSON object	Convert a string to a JSON object.
stringifyJson	Specify any JSON object.	String representation of JSON	Convert a JSON object to a string.
base64.encode	Specify a string to be converted to BASE64.	String encoded to BASE64	Encode into BASE64.

Method Name	Argument	Return Value	Description
base64.decode	Specify a BASE64 string.	String decoded from BASE64	Decode from BASE64.
http.call	JSON object of the request	JSON object of the response	Perform http or https request and return a response.
storage.restCall	JSON object of the request	JSON object of the response	Perform an http or https request and return a response. Use this method when you run the Configuration Manager REST API. Compared with tbe http.call method, the specifications of the accept header, timeout period, and log output are different. For details about the differences, see the separate table below.
http.toRawHeader	JSON object of the header	Header string	Return the header in the form of string. To the argument, specify the JSON object where the key and its value is set.
http.defaultErrorHandler	Error JSON object of the request JSON object of the response	None	Throw HttpError instance.
http.handleCall	1. httpCall method 2. JSON object of the request 3. Method to call when the http call is successful (The result status code is 200 or more and less than 400).	None	Call the httpCall method of the first argument by specifying the request of the second argument. If the response of the result is 200 or more and less than 400, call the third argument, and in the case of other response codes, call the fourth argument. If the http call fails and the http response is not received, call the fifth argument.

Method Name	Argument	Return Value	Description
	4. Method to call when the http call fails (The result status code is less than 200 or more than 400). **5.** Method to call when the http call fails and the http response is not received. **6.** Object with retry conditions (For members of this object, see the separate table below.)		The JSON objects of the request, response, and error when calling the httpCall method in the first argument can be called with the variables "req", "resp", "err", respectively, and you can use them as the argument of the function object in the third through fifth arguments. If auto.util.storage.retrySettings is specified as the sixth argument, the retry is performed with the following retry count and interval when the HTTP response meets the following conditions. ▪ Retry conditions: One of the following conditions is met: (1) Status code is 503. (2) Status code is 500 & Content-Type of response headers is text/html ▪ Retry count: 130 (30 when called from the external resource provider) ▪ Retry interval: 30 seconds (10 seconds when called from the external provider)
http.getHeaderValue	Specify headers of the JSON object of the response by string type.	String value of the header key name.	Gets the value of the response header. If there are duplicate response header keys, the first header found is returned. If a nonexistent key is specified, an empty character is returned.

Method Name	Argument	Return Value	Description
	Specify the key name of the header to acquire by string type. The value to be specified is case-insensitive.		
http.buildQuery	Specify the query parameters in the Map format.	Output query parameters in the string format.	Builds query parameters and perform URL encoding. For a code sample, see Sample code (Running the Configuration Manager REST API) (on page 248)
env.getServiceTemplate	-	Service template object	Get the object of the service template.
env.getService	-	Service object	Get the object of the service.
env.getWebServiceConnections	Category name of the Web Service Connection. If not specified, all categories are obtained.	A list of JSON objects (except for passwords) of Web Service Connections with the same category name as the one specified as an input parameter.	Get a list of Web Service Connections. Web Service Connections that are accessible by the service or task are obtained by considering the defined relation between the Service, Service Group, Infrastructure Group, and Web Service Connection.

The following members are available for the JSON object of the request:

Member	Type	Description
requestUrl	String	[If not using Web Service Connection] Specify the request URL starting with http or https. [If using Web Service Connection] Specify the part from the "/" after the host name of the request URL to the en. Example: In the case of `http://host:port/folder`, specify "/Folder".
requestMethod	String	HTTP request method. Specify the following value as the string. ▪ GET ▪ POST ▪ PUT ▪ DELETE ▪ PATCH (Supported only when "none" or "basic" is specified for authScheme.)
requestHeaders	String	Request header. Assume that the return value of the http.toRawHeader method is set. [When using Web Service Connection] The following values can be used as pad characters for user ID and password. - ${connection.username} - ${connection.password}
requestBody	String	Request body
authScheme	String	Specify one of the following values: ▪ none ▪ basic ▪ digest ▪ negotiate

Member	Type	Description
		If you specify the Authorization header for the request in the JavaScript code of the JavaScript Plug-in or external resource provider, specify “none”.
productName	String	Specify Web Service Connection category. Make sure to specify if using Web Service Connection.
connectionName	String	Specify Web Service Connection name. Make sure to specify if using Web Service Connection.
userName	String	Specify the user name to authenticate to the destination. Do not specify if using Web Service Connection.
password	String	Specify the password to authenticate to the destination.
useProxy	boolean	Specify whether to use a proxy (true/false).
proxyHost	String	Specify the host name or IP address of the proxy server.
proxyPort	int	Specify the port number of the proxy server.
proxyAuthScheme	String	If you specify "true" for useProxy, specify one of the following values: ▪ none ▪ basic ▪ digest
proxyUserName	String	Specify the user name if the authentication is required at the proxy server.

Member	Type	Description
proxyPassword	String	Specify the password if the authentication is required at the proxy server.
conntectionTimeout	int	Specify the timeout value in seconds (0-3600) when establishing an HTTP/ HTTPS connection. If “0” is specified, no timeout occurs.
readTimeout	int	Specify the timeout value in seconds (0-86400) for reading data after an HTTP/ HTTPS connection is established. If “0” is specified, no timeout occurs.
outputLog	boolean	Specify whether or not to output the response body to the task log. If anything other than true/ false is specified, the default value "true" is used. Note that it can be specified only when the auto.util.storage.restCall method is run (it is ignored if specified in another method).

The following table shows the allowable response members:

Member	Type	Description
responseHeaders	String	Response header
responseStatusCode	int	Response code
responseStatusMessage	String	Response message
responseBody	String	Response body

The following members are available for the object with retry conditions:

Member	Type	Description
retryCondition	Function object	Function that returns the retry necessity (true/false) based on the response of the API that was run immediately before. If true, the retry is performed, and if false, the retry is not performed. For a sample code, see "Sample code (Retry settings)" below.
maxRetryCount	int	Retry count. Specify an integer between 0 and 1000.
retryWaitTime	int	Retry interval (in seconds). Specify an integer between 0 and 3600.

The following table shows the specification differences between the `http.call` and `storage.restCall` methods.

Item	http.call	storage.restCall
Accept header	application/json	*/*
Timeout period	Connection: 60 seconds (1 minutes) Read: 600 seconds (10 minutes)	Connection: 600 seconds (10 minutes) Read: 10800 seconds (180 minutes)
Log output	Logs for the request URL, response code, and response body are not output to the task log.	Logs including the request URL, response code, and response body are output to the task log.

Sample code (scriptBody)

JavaScript sample code that can be specified in the plug-in property JavaScript body is as follows:

```
function(serviceProperties, pluginProperties, arg0, arg1, arg2) {
    var obj = new Object();
    print("[Debug] Function begin.");

    obj.mem1 = arg0;
    obj.mem2 = arg1;
```

```
    if (arg2 == "") {
        pluginProperties["notify"] = 999;
        pluginProperties["out1"] = "NOTE!: The arg2 is EMPTY.";
    } else {
        obj.mem3 = arg2;
        obj.status = "success";
        pluginProperties["out1"] = "Finished successfully.";
    }

    print("[Debug] Function end.");

    return obj;
}
```

As described in the previous table, there are two ways of obtaining arg0, arg1,... where each can provide a different value. For pluginProperties, the arguments are obtained as a string but when obtained from the argument arg0, arg1,..., the evaluated value can be obtained from JavaScript. The example that shows the difference between the case of obtaining the input property of the plug-in from pluginProperties and the case of obtaining it from arg0, arg1,... is as follows.

Suppose that the following sample code is run by specifying 3 to arg0 and 5 to arg1.

```
function(serviceProperties, pluginProperties, arg0, arg1) {
    pluginProperties["out0"] = pluginProperties["arg0"] + pluginProperties["arg1"];
    pluginProperties["out1"] = arg0 + arg1;
}
```

When this sample code is run, "35" is set to out0, and "8.0" is set to out1. From pluginProperties, the arguments are handled as strings of "3" and "5" respectively, while the "+" symbol is handled as the string concatenation. In addition, for arg0 and arg1, they become 3.0 and 5.0 respectively, while the "+" symbol is handled as the addition function.

The following sample shows how to get the pool list from the Platform REST API server included in the storage system by referring to the connection information provided by Web Service Connection, and sets to the output property `out0`. As a prerequisite, the Web Service Connection whose category is StorageSystem" and whose name is "storage1" must already be registered.

```
function getPools(productname, connectionname) {
  var respBody = null;
  var request = {
    requestMethod:'GET',
    requestUrl:'/ConfigurationManager/v1/objects/pools',
    requestHeaders:auto.util.http.toRawHeader({
      'Accept':'application/json',
      'Accept-Language':'en',
      'Content-Type':'application/json',
    }),
    authScheme:'basic',
```

```
    connectionName: connectionname,
    productName: productname,
  };
  auto.util.http.handleCall(auto.util.http.call, request,
    function(resp, req) {
      respBody = resp.responseBody;
    }, function(resp, req) {
      auto.util.http.defaultErrorHandler(null, req, resp);
    }, function(err, req) {
      auto.util.http.defaultErrorHandler(err, req);
    }
  );
  return JSON.parse(respBody);
}
function fn(serviceProperties, pluginProperties, arg0, arg1, arg2, arg3, arg4, arg5,
arg6, arg7, arg8, arg9) {
  var data = getPools('StorageSystem', 'storage1').data;
  var poolList = [];
  _.each(data, function(d){
    poolList.push({
      poolId: d.poolId,
      poolStatus: d.poolStatus,
      availableVolumeCapacity: d.availableVolumeCapacity
    });
  });
  pluginProperties.out0 = JSON.stringify(poolList, null, 2);
}
```

The following sample shows how to get the pool list from the Platform REST API server included in the storage system without referring to the connection information of Web Service Connection, and sets to the output property `out0`. As a prerequisite, the host name of Platform REST API server must be "host", the logon user name is "user", and the password is "password".

```
function getPools() {
  var respBody = null;
  var request = {
    requestMethod:'GET',
    requestUrl:'https://host/ConfigurationManager/v1/objects/pools',
    requestHeaders:auto.util.http.toRawHeader({
      'Accept':'application/json',
      'Accept-Language':'en',
      'Content-Type':'application/json',
    }),
    authScheme:'basic',
    userName:'user',
    password:'password',
    useProxy:'false'
  };
  auto.util.http.handleCall(auto.util.http.call, request,
```

```
    function(resp, req) {
      respBody = resp.responseBody;
    }, function(resp, req) {
      auto.util.http.defaultErrorHandler(null, req, resp);
    }, function(err, req) {
      auto.util.http.defaultErrorHandler(err, req);
    }
  );
  return JSON.parse(respBody);
}
function fn(serviceProperties, pluginProperties, arg0, arg1, arg2, arg3, arg4, arg5,
arg6, arg7, arg8, arg9) {
  var data = getPools().data;
  var poolList = [];
  _.each(data, function(d){
    poolList.push({
      poolId: d.poolId,
      poolStatus: d.poolStatus,
      availableVolumeCapacity: d.availableVolumeCapacity
    });
  });
  pluginProperties.out0 = JSON.stringify(poolList, null, 2);
}
```

Sample code (scriptBody/importedScript)

JavaScript sample code that can be specified in the plug-in property scriptBody/ importedScript is as follows:

```
-----------------------------------------
(scriptBody)
-----------------------------------------
function fn(serviceProperties, pluginProperties, arg0, arg1, arg2, arg3, arg4, arg5,
arg6, arg7, arg8, arg9) {

  hoge(CNST);

}
-----------------------------------------
(importedScript)
-----------------------------------------
var CNST = "hoge";

function hoge(a){
  print(a + " from common js!");
}
-----------------------------------------
```

Sample code (Retry settings)

JavaScript sample code in which the custom retry condition is specified as the sixth argument of http.handleCall.

```
var retrySettings = {};
retrySettings.retryCondition = function (response) {return
response.responseStatusCode===503;};  // Retry when the response is 503
retrySettings.maxRetryCount = 10;  // Retry up to 10 times
retrySettings.retryWaitTime = 60;  // Retry every 60 seconds

auto.util.http.handleCall(auto.util.storage.restCall, request, successHandler,
serverErrorHandler, clientErrorHandler, retrySettings);
```

Sample code (Running the Configuration Manager REST API)

The following shows a sample code when running the Configuration Manager REST API to obtain the pool information.

```
function fn(serviceProperties, pluginProperties, arg0, arg1, arg2, arg3, arg4, arg5,
arg6, arg7, arg8, arg9) {
  var storageDeviceId = <storageDeviceId>;

  //Step 1. Generate a method to run REST API to Configuration Manager.
  var configurationManagerCall = function(request) {
    var respBody = null;
    auto.util.http.handleCall(auto.util.storage.restCall, request,
      function(resp, req) {
        respBody = auto.util.parseJson(resp.responseBody);
      }, function(resp, req) {
        auto.util.http.defaultErrorHandler(null, req, resp);
      }, function(err, req) {
        auto.util.http.defaultErrorHandler(err, req);
      }, auto.util.storage.retrySettings
    );
    return respBody;
  };

  //Step 2. Get accessible Web Service Connections by specifying a category name.
  var wsc = auto.util.env.getWebServiceConnections("ConfigurationManager");

  //Step 3. Get Session
  //You can pass arguments by using the plug-in input properties or you can specify
them directly in a script. For example, you can pass the device ID through the query
parameters (queryParamMap).
  var request = {
    "requestMethod": "POST",
    "requestUrl": "/ConfigurationManager/v1/objects/storages/" + storageDeviceId + "/
sessions",
    "requestHeaders": auto.util.http.toRawHeader({}),
    "authScheme": "basic",
```

```
    "connectionName": wsc[0].name,
    "productName": wsc[0].productName
  };
  var responseBody = configurationManagerCall(request);
  var token = responseBody.token;
  var sId = responseBody.sessionId;

  //Step 4. Call the API that is associated with your use case based on the session
obtained in Step 3.
//When specifying the Authorization header for the request in the user program,
specify "none" for authScheme.
  var headers = {
    "Authorization": "Session " + token,
  };
  var queryMap = {
    "poolType" : "DP"
  };
  var queryStr = auto.util.http.buildQuery(queryMap);
  request = {
    "requestMethod": "GET",
    "requestUrl": "/ConfigurationManager/v1/objects/storages/" + storageDeviceId + "/
pools" + queryStr,
    "requestHeaders": auto.util.http.toRawHeader(headers),
    "authScheme": "none",
    "connectionName": wsc[0].name,
    "productName": wsc[0].productName
  };
  var ret = configurationManagerCall(request);

  //Final Step. Set the required information for the output property of the plug-in
or return value.
  pluginProperties.out0 = ret.data[0].poolId;
  pluginProperties.out1 = ret.data[0].poolName;
  return ret;
}
```

Python Plug-in

The Python plug-in is a component that runs Python scripts on the Ops Center Automator host. The Python interpreter must be installed on the Ops Center Automator host before it can be used, because the Python interpreter is not installed by Ops Center Automator. To use this plug-in in a cluster environment, the Python interpreter must be installed on both the active and standby systems. The Python plug-in does not support a virtual Python environment.

Supported versions of Python

Version 3.x series

Return codes

The Python Plug-in generates the following return codes:

Return Code	Description
0	Ended normally.
1	Python interpreter failed.
2	Python script failed.
3	Python script timed out.
80	Task has stopped.
127	Another error has occurred.

Property list

The following properties are available for the Python Plug-in:

Property key	Property name	Description	Default value	I/O type	Required
pythonInterpreterPath	Python interpreter path	Specifies the path to the Python interpreter that executes the script	python	Input	true
scriptBody	Script body	Specifies the Python code strings.	--	Input	true
importedScript	Imported script	Specifies methods and constants (code string of Python) to be used in common with other Python Plug-ins placed on the same service template.	--	Input	false

Property key	Property name	Description	Default value	I/O type	Required
		The following items can be used in the script of importedScript: ▪ Functions: log() function. Same as the log() function available at scriptBody. ▪ Environment variables: Same as the environment variables available at scriptBody.			
webServiceConnectionCategory	Web Service Connection Category	Specify the Web Service Connection category.	--	Input	false
webServiceConnectionName	Web Service Connection Name	Specify the Web Service Connection name.	--	Input	false
timeout	Timeout	Specifies the timeout time (in seconds) for the specified script.	300	Input	false
in*N*	Input(*N*)	Specifies an argument to be passed to the script.	--	Input	false
standardOutput	Standard output	Outputs the standard output of the specified script as a character string	--	Output	false
standardErrorOutput	Standard error output	Outputs the standard error output of the specified script as a character string	--	Output	false
out*N*	Output(*N*)	Outputs the value specified for the argument of the "out*N*" function in the specified script	--	Output	false
N: An integer in the range from 0 to 9					

You specify the plug-in input/output properties in the property list. Combinations of service property values, reserved property values, and literal characters can be used for the input properties.

Variables and functions that can be used in the script

The following variables and functions can be used in the script.

Category	Name	Description		
Variable	in*N*	The value specified for the input property (in*N*) is set. The value is interpreted as a character string even if the value is specified as an array or in JSON format.		
Function	out*N* (String Value)	The value passed as the argument of the function is output to the component output property (out*N*).		
Function	log (String Value)	You can output any strings to the tasklog. In this case, choose a log level by adding a specific prefix to the beginning of the string.		
		Prefix	[Severe]	Outputs as log level 0
			[Information]	Outputs as log level 10
			[Fine]	Outputs as log level 20
			[Finer]	Outputs as log level 30
			[Debug]	Outputs as log level 40
			(No prefix)	Same as the prefix [Information]
N: An integer in the range from 0 to 9				

Note that if the following import lines defined by default are deleted from the script, the variables and functions mentioned previously cannot be used.

```
from dnaplugin import in0, in1, in2, in3, in4, in5, in6, in7, in8, in9
from dnaplugin import out0, out1, out2, out3, out4, out5, out6, out7, out8, out9
from dnaplugin import log
```

Environment variables that can be referenced from the script

Specify a value for the environment variables when executing the script. You can get the values for the following environment variables in the os.environ.get(*key-name*) or os.environ[*key-name*] format.

Environment variable	Description	Format
PLUGIN_PROPERTIES	Property for the Python plug-in	JSON format {*property-name*:*value*, ...}
SERVICE_TEMPLATE_ID	ID of the service template to which the Python plug-in belongs	Numerical value
SERVICE_ID	ID of the service running the Python plug-in	Numerical value
SERVICE_TEMPLATE	Information about the service template to which the Python plug-in belongs	JSON format {*service-template-attribute*:*value*, ...}
SERVICE	Information about the service running the Python Plug-in	JSON format {*service-attribute*:*value*, ...}
STORAGE_PROFILES	Information about the Storage Profile	JSON format [{ *StorageProfile-attribute*:*value, ...* }, ...]
WEB_SERVICE_CONNECTIONS	Settings information for the Web Service Connection. This corresponds to the specified input properties ("Web Service Connection Category" and "Web Service Connection Name") seen in the following table.	JSON format [{ *WebServiceConnection-attribute*:*value, ...* }, ...]

Input properties		Reference information
Web Service Connection Category	**Web Service Connection Name**	
Value is specified. (Y)	Value is specified. (Y)	Web Service Connection information that coincides with the specified Category and Name
Value is specified. (Y)	Value is not specified. (N)	Web Service Connection information that coincides with the specified Category
Value is not specified. (N)	Value is specified. (Y)	None

Input properties		
Web Service Connection Category	Web Service Connection Name	Reference information
Value is not specified. (N)	Value is not specified. (N)	None

Sample code (scriptBody/importedScript)

The following examples show the sample code of the Python Plug-in that can be specified in the plug-in property scriptBody/importedScript:

```
-----------------------------------------
(scriptBody)
-----------------------------------------
# -*- coding: utf-8 -*-
import os
from dnaplugin import in0, in1, in2, in3, in4, in5, in6, in7, in8, in9
from dnaplugin import out0, out1, out2, out3, out4, out5, out6, out7, out8, out9
from dnaplugin import log
from dnaplugin_imported_script import *

hoge(CNST)
-----------------------------------------
(importedScript)
-----------------------------------------
from dnaplugin import log

def hoge(a):
  log(a + ' from common py!')

CNST = 'hoge'
-----------------------------------------
```

Web Client Plug-in

The Web Client Plug-in sends and receives HTTP request and response messages. When requested, it accesses the server by using a proxy and completes server and proxy authentication.

Function

This plug-in sends and receives HTTP request and response messages and consists of the following functions:

- Supports HTTP/HTTPS 1.1.
- Generates an HTTP request message based on the input properties and receives HTTP response messages as output properties.

The following table shows the relationships between HTTP request messages and their corresponding input properties.

Element		Input Property
Request line	Method	requestMethod
	URI	requestUrl
	HTTP/version	-
Header		requestHeaders
Body		requestBody

The following table shows the relationships between HTTP response messages and their corresponding output properties.

Element		Output Property
Status line	HTTP/version	-
	Status code	responseStatusCode
	Status message	responseStatusMessage
Header		responseHeaders
Body		responseBody

Prerequisites

- To use HTTPS communication, import the destination server certificate or a root certificate that authenticates the server certificate in JRE truststore on the client (Ops Center Automator installation server).

Return codes

The Web Client Plug-in generates the following return codes:

Return Code	Description
0	Ended normally.
1	A status code other than Success is returned for an HTTP response message.
70	Failed to connect to the remote host.
77	Failed to resolve the host name for the remote host.

Return Code	Description
80	Task execution has stopped.
86	An incorrect property value has been specified.
90	A data transmission failure occurred after the connection was established.
91	The size of the HTTP response message exceeds the upper limit value of the system.
127	Another unspecified type of error has occurred.

Property list

The properties available for the Web Client Plug-in. See details in Table 13 Relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in (on page 262)

Property key	Property name	Description	I/O type
webServiceConnectionCategory	Web Service Connection Category	When referring to the information of Web Service Connection and using it as an input value of Web Client Plug-in, specify the category of Web Service Connection.	Input
webServiceConnectionName	Web Service Connection Name	When referring to the information of Web Service Connection and using it as an input value of Web Client Plug-in, specify the name of Web Service Connection.	Input
requestMethod R	Method	Specify the HTTP method as follows: ▪ GET ▪ POST ▪ PUT ▪ DELETE The default value is GET.	Input

Property key	Property name	Description	I/O type
requestUrl R	Request URL	Specify the resource URL to which the HTTP request is sent, and a query parameter. The URL-encoded value specified as the input property value is used as is. See Table 13 Relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in (on page 262) for more details.	Input
requestHeaders	Request Headers	Specify the HTTP request header in raw format. Use the Content-Type header and the charset parameter to specify the request body format and character set.	Input
requestBody	Request Body	Specify the HTTP request body in the raw format. Use the format specified in the Content-Type header.	Input
webAuth R	Server Authentication Scheme	Specify the server authentication type: ▪ none -- No authentication ▪ basic -- Basic authentication ▪ digest -- Digest authentication ▪ negotiate -- Negotiate authentication Specify "none" for the authentication header in `requestHeaders`.	Input

Property key	Property name	Description	I/O type
webUsername	Server Authentication Username	Specify the user name used for server authentication, using a maximum of 256 characters. Not valid when the webAuth property key is set to none. See Table 13 Relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in (on page 262) for more details.	Input
webPassword	Server Authentication Password	Specify the password used for server authentication, using a maximum of 256 characters. Not valid when the webAuth property key is set to none. See Table 13 Relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in (on page 262) for more details.	Input
useProxy [R]	Use Proxy Server	Set this to true when a proxy connection is required. See Table 13 Relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in (on page 262) for more details. The default value is false.	Input
proxyHostname	Proxy Hostname	Specify the proxy host name or IP address, using a maximum of 256 characters. Not valid when the useProxy property key is set to false. See Table 13 Relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in (on page 262) for more details.	Input

Property key	Property name	Description	I/O type
proxyPort	Proxy Port Number	Specify the proxy port number, using 1-65535. Not valid when the useProxy property key is set to false. See Table 13 Relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in (on page 262) for more details. The default value is 8080.	Input
proxyAuth [R]	Proxy Server Authentication Scheme	Specifies the proxy authentication type. The following authentication functions are supported: ▪ none -- No authentication ▪ basic -- Basic authentication ▪ digest -- Digest authentication Specify "none" for the authentication header in requestHeaders. See Table 13 Relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in (on page 262) for more details. The default value is none.	Input
proxyUsername	Proxy Username	Specify the user name used for proxy authentication, using a maximum of 256 characters. Not valid when the useProxy property key is set to false or the proxyAuth property key is set to none. See Table 13 Relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in (on page 262) for more details.	Input

Property key	Property name	Description	I/O type
proxyPassword	Proxy Password	Specify the password used for proxy authentication, using a maximum of 256 characters. Not valid when the useProxy property key is set to false or the proxyAuth property key is set to none. See Table 13 Relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in (on page 262) for more details.	Input
responseStatusCode	Status Code	Outputs the status code of the HTTP response message. When "3xx Redirect" is returned, automatic tracking is activated.	Output
responseStatusMessage	Status Message	Outputs the status message of the HTTP response message.	Output
responseHeaders	Response Headers	Outputs the header information of the HTTP response message.	Output
responseBody	Response Body	Outputs the body information of the HTTP response message. Make sure that the size of responseBody does not exceed 30 MB. If the size of responseBody exceeds 30 MB, the value is truncated.	Output
R: Required			

Note: When using Basic authentication, a request without authentication returns the 401 response code and a "WWW-Authenticate:Basic" header indicating that authentication is required. If "WWW-Authenticate:Basic" is not returned, the authentication token with Base64-converted "Username:Password" is added to the request header. Authentication by the preemptive method is not supported because Digest authentication requires values such as a nonce value to be returned from the server.

The setting value of Web Service Connection can be referred to as a value of the input property of the Web Client Plug-in. In this case, the corresponding input property is overwritten by the setting value of the Web Service Connection.

Whether to refer to the setting value of Web Service Connection is determined by the input value of webServiceConnectionCategory and webServiceConnectionName.

Input Value of webServiceConnectionCategory	Input Value of webServiceConnectionName	Behavior of Web Client Plug-in
Exist	Exist	See the value of Web Service Connection. If no matching Web Service Connection is found, it becomes a run-time error.
Exist	Not exist	Error (tried to refer to Web Service Connection, but did not find the matching one).
Not exist	Exist	Error (tried to refer to Web Service Connection, but did not find the matching one).
Not exist	Not exist	Do not refer to the value of Web Service Connection.

When referring to the setting value of WebServiceConnection, the URL (property name: requestUrl) is assembled dynamically at the run time. As the input value of the property, you must specify the part after "/" after the host name. The part before "/" is assembled from the setting value of Web Service Connection. In the case of the string, where the value specified in requestUrl does not begin with "/", it generates a run-time error.

Example: In the case of "http://host:port/Folder/", specify "/Folder/" as the input value. The following table shows the relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in:

Table 13 Relationship between the value of the Web Service Connection and the input value of the Web Client Plug-in

Item of the Web Service Connection	Input Property of Web Client Plug-in	Remark
IP Address/Host Name	Part of requestUrl	host part of "http://host:port/Folder/"
Protocol	Part of requestUrl	http part of "http://host:port/Folder/"
Port	Part of requestUrl	port part of "http://host:port/Folder/"
User ID	webUsername	Output the message to the task logs that overwriting the input value if it was set.
Password	webPassword	Output the message to the task log that overwriting the input value if it was set.
Use Proxy Server	useProxy	-
IP Address/Host Name in Use Proxy Server	proxyHostname	Output the message to the task log that overwriting the input value if it was set.
Port in Use Proxy Server	proxyPort	Output the message to the task log that overwriting the input value if it was set.
Authentication Type in Use Proxy Server	proxyAuth	-
User ID in Use Proxy Server	proxyUsername	Output the message to the task log that overwriting the input value if it was set.
Password in Use Proxy Server	proxyPassword	Output the message to the task log that overwriting the input value if it was set.

Supported Headers

Headers are transmitted and received in their raw format. The following default headers are supported:

Header	Value
Accept	application/json
Accept-Language	en
Content-Type (Only when POST or PUT is specified as the method.)	application/json
Cache-Control	no-cache
Pragma	no-cache
User-Agent	client-software-name, version
Host	destination-host-name, port-number
Connection	keep-alive

The following headers have a special behavior:

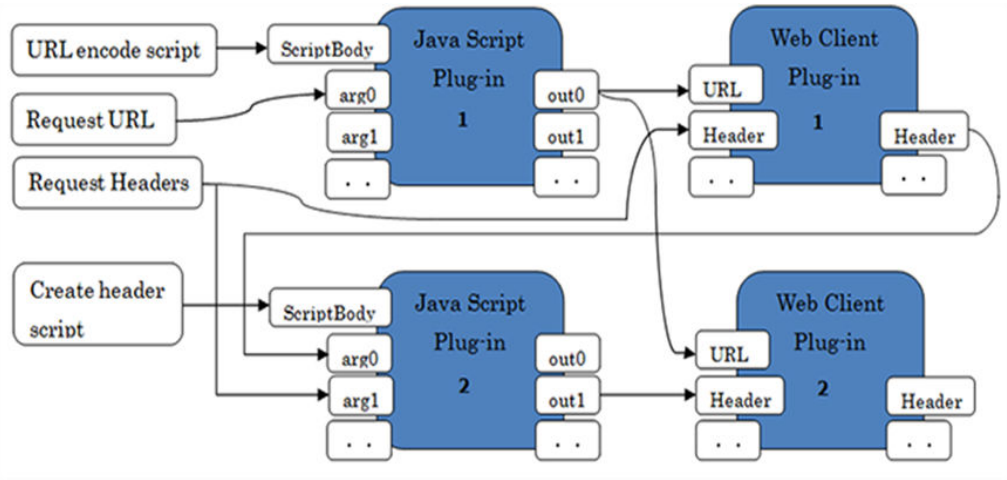
Header	Behavior
`Charset` parameter in the Content-Type header	Request: The body character set is converted by using the value specified in the Charset parameter of the Content-Type header. When no value is specified, it is converted to UTF-8.
	Response: The body character set is interpreted according to the value specified in the Charset parameter of the Content-Type header. When a charset parameter is not returned, it is interpreted as UTF-8.
Content-Encoding	When the Content-Encoding header is returned, the body is extended. The following encoding formats are supported: ▪ gzip ▪ deflate

Connection timeout value settings

When dealing with any HTTP/HTTPS communication problems that might occur when connecting to the destination, you should configure the connection timeout key value (`plugin.http.connect.timeout`) so it is obvious when a problem occurs with the connection. The connection timeout value is specified through the key name (`config_user.properties`) in the properties file under *`automation-software-installation-folder`*`\conf`.

Linkage with the JavaScript Plug-in

Because the Web Client Plug-in does not rewrite input and output property values, linkage with the JavaScript Plug-in is effective when values must be rewritten. Following is an example of linkage with the JavaScript Plug-in to extract the URL encoding and the SSO authentication token from the server response.



The following table shows the flow linkage with the JavaScript Plug-in.

Plug-in	Flow	Description
JavaScript Plug-in 1	Input	URL entered by the user. Script that runs URL encoding.
	Execute	Runs URL encoding.
	Output	URL-encoded URL.
Web Client Plug-in 1	Input	URL-encoded URL. Other information entered by the user (such as the header).
	Execute	Generates and sends the HTTP request. Receives and analyzes the HTTP response.
	Output	Outputs the HTTP response elements (such as the header).

Plug-in	Flow	Description
JavaScript Plug-in 2	Input	URL-encoded URL. Reconfigured header. Other information entered by the user.
	Execute	Generates and sends the HTTP request. Receives and analyzes the HTTP response.
	Output	Outputs the HTTP response elements.

Prerequisites for using Negotiate authentication

Negotiate authentication uses Kerberos v5 authentication, which needs the following configuration files:

File Name	Description	Edited by user
Kerberos configuration file (krb5.conf)	Kerberos configuration in the user environment.	Required
Login configuration file (login.conf)	Specifies the authentication technology to be used.	Not required

The following code shows that the Kerberos configuration file is in *automation-software-installation-folder*\conf\plugin. Edit the code, especially the italicized characters, as needed for the user environment.

```
[libdefaults] // Default value for Kerberos authentication
 default_realm = EXAMPLE.COM // Default realm for Kerberos authentication
 udp_preference_limit = 1

[realms] // KDC setting for each Kerberos realm (you can define settings for multiple
realms)
 EXAMPLE.COM = {
  kdc = kdc.example.com // KDC host name
 }

[domain_realm] // Maps the Active Directory domain to the Kerberos realm
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

Note:

- Realms are case-sensitive. Because uppercase letters are conventionally used, use uppercase to specify realms (lowercase letters cannot be used).
- Although Kerberos authentication uses UDP by default, it uses TCP for lager messages. Because TGT requests that use UDP will fail to link with Active Directory, use udp_preference_limit=1 so that TCP is used.
- Negotiate authentication is available only if Windows is the OS of the management server on which Ops Center Automator is installed.
- Only the computer account authentication is supported for Negotiate authentication.

Login Configuration File

The following login configuration file is in *automation-software-installation-folder*\conf\plugin.

```
com.sun.security.jgss.krb5.initiate {
  com.sun.security.auth.module.Krb5LoginModule required useTicketCache=true
doNotPrompt=false refreshKrb5Config=true; // Specifies the authentication
technology and options to be used
};
```

Enter the following in *Common-Component-installation-folder*\uCPSB11\CC\server\usrconf\ejb\AutomationWebService\usrconf.cfg so that the Kerberos configuration file and the login configuration file are referenced.

Note: The settings in the usrconf.cfg file are reset at the following times. Reconfigure the file after:

- Performing an upgrade installation or overwrite installation of Ops Center Automator
- Performing a new installation, upgrade installation, overwrite installation, or removal of a Common Component product
- Changing the performance mode with the **changemode** command

```
add.jvm.arg=-Djava.security.krb5.conf=automation-software-installation-folder/conf/
plugin/krb5.conf
add.jvm.arg=-Djava.security.auth.login.config=automation-software-installation-folder/
conf/plugin/login.conf add.jvm.arg=-Djavax.security.auth.useSubjectCredsOnly=false
```

Appendix C: Notices

This software product includes the following redistributable software.

Notices

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

Portions of this software were developed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign.

This product includes software developed by the University of California, Berkeley and its contributors.

This software contains code derived from the RSA Data Security Inc. MD5 Message-Digest Algorithm, including various modifications by Spyglass Inc., Carnegie Mellon University, and Bell Communications Research, Inc (Bellcore).

Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England. The original software is available from ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/

1. This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)

2. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)

3. This product includes software written by Tim Hudson (tjh@cryptsoft.com)

4. This product includes the OpenSSL Toolkit software used under OpenSSL License and Original SSLeay License. OpenSSL License and Original SSLeay License are as follow:

LICENSE ISSUES

==========

The OpenSSL toolkit stays under a double license, i.e. both the conditions of

the OpenSSL License and the original SSLeay license apply to the toolkit.

See below for the actual license texts.

OpenSSL License

/* ==

* Copyright (c) 1998-2019 The OpenSSL Project. All rights reserved.

*

* Redistribution and use in source and binary forms, with or without

* modification, are permitted provided that the following conditions

* are met:

*

* 1. Redistributions of source code must retain the above copyright

* notice, this list of conditions and the following disclaimer.

*

* 2. Redistributions in binary form must reproduce the above copyright

* notice, this list of conditions and the following disclaimer in

* the documentation and/or other materials provided with the

* distribution.

*

* 3. All advertising materials mentioning features or use of this

* software must display the following acknowledgment:

* "This product includes software developed by the OpenSSL Project
* for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
*
* 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
* endorse or promote products derived from this software without
* prior written permission. For written permission, please contact
* openssl-core@openssl.org.
*
* 5. Products derived from this software may not be called "OpenSSL"
* nor may "OpenSSL" appear in their names without prior written
* permission of the OpenSSL Project.
*
* 6. Redistributions of any form whatsoever must retain the following
* acknowledgment:
* "This product includes software developed by the OpenSSL Project
* for use in the OpenSSL Toolkit (http://www.openssl.org/)"
*
* THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
* ==
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/

Original SSLeay License

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms

* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
* notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
* documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
* must display the following acknowledgement:
* "This product includes cryptographic software written by
* Eric Young (eay@cryptsoft.com)"
* The word 'cryptographic' can be left out if the rouines from the library
* being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
* the apps directory (application code) you must include an acknowledgement:
* "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/

This product includes the OpenSSL library.

The OpenSSL library is licensed under Apache License, Version 2.0.

https://www.apache.org/licenses/LICENSE-2.0

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

This product includes software developed by IAIK of Graz University of Technology.

This product includes software developed by Daisuke Okajima and Kohsuke Kawaguchi (http://relaxngcc.sf.net/).

This product includes software developed by the Java Apache Project for use in the Apache JServ servlet engine project (http://java.apache.org/).

This product includes software developed by Andy Clark.

Java is a registered trademark of Oracle and/or its affiliates.

Other company and product names mentioned in this document may be the trademarks of their respective owners.

Hitachi Vantara

Corporate Headquarters
2535 Augustine Drive
Santa Clara, CA 95054 USA

HitachiVantara.com/contact