

**JP1 Version 11**

**JP1/Automatic Operation Service Template  
Developer's Guide**

**3021-3-A90-40(E)**

## Notices

### ■ Relevant program products

P-2A2C-E1BL JP1/Automatic Operation 11-51 (for Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016)

The above product includes the following:

- P-CC2A2C-EABL JP1/Automatic Operation - Server 11-51 (for Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016)
- P-CC2A2C-EBBL JP1/Automatic Operation - Contents 11-51 (for Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016)

P-2A2C-E3BL JP1/Automatic Operation Content Pack 11-51 (for Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016)

P-822C-E1BL JP1/Automatic Operation 11-51 (for Linux 6 (x64), Linux 7, Oracle Linux 6 (x64), Oracle Linux 7, CentOS 6 (x64), CentOS 7, SUSE Linux 12)

The above product includes the following:

- P-CC822C-EABL JP1/Automatic Operation - Server 11-51 (for Linux 6 (x64), Linux 7, Oracle Linux 6 (x64), Oracle Linux 7, CentOS 6 (x64), CentOS 7, SUSE Linux 12)
- P-CC822C-EBBL JP1/Automatic Operation - Contents 11-51 (for Linux 6 (x64), Linux 7, Oracle Linux 6 (x64), Oracle Linux 7, CentOS 6 (x64), CentOS 7, SUSE Linux 12)

P-822C-E3BL JP1/Automatic Operation Content Pack 11-51 (for Linux 6 (x64), Linux 7, Oracle Linux 6 (x64), Oracle Linux 7, CentOS 6 (x64), CentOS 7, SUSE Linux 12)

### ■ Trademarks

HITACHI, HiRDB, JP1 are either trademarks or registered trademarks of Hitachi, Ltd. in Japan and other countries. IBM, AIX are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Intel is a trademark of Intel Corporation in the U.S. and/or other countries.

Internet Explorer is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

Linux(R) is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft and Hyper-V are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

RSA and BSAFE are either registered trademarks or trademarks of EMC Corporation in the United States and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Windows is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Windows Server is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Other company and product names mentioned in this document may be the trademarks of their respective owners.

This product includes software developed by Andy Clark.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

This product includes software developed by Ben Laurie for use in the Apache-SSL HTTP server project.

This product includes software developed by Daisuke Okajima and Kohsuke Kawaguchi (<http://relaxngcc.sf.net/>).

This product includes software developed by IAIK of Graz University of Technology.

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 Ralf S. Engelschall <[rse@engelschall.com](mailto:rse@engelschall.com)> for use in the mod\_ssl project (<http://www.modssl.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/>



JP1/Automatic Operation includes RSA BSAFE(R) Cryptographic software of EMC Corporation.

Java is a registered trademark of Oracle and/or its affiliates.

**HITACHI**  
Inspire the Next

 Hitachi, Ltd.



## ■ Issued

Apr. 2018: 3021-3-A90-40(E)

## ■ Copyright

All Rights Reserved. Copyright (C) 2016, 2018, Hitachi, Ltd.

## Summary of amendments

The following table lists changes in this manual (3021-3-A90-40(E)) and product changes related to this manual.

Changes	Location
In the <b>Specify Component Input Property Mapping Parameters</b> dialog box or the <b>Specify Component Output Property Mapping Parameters</b> dialog box, by selecting the <b>View Property</b> radio button, you can now display only the properties that can be mapped. As such, the relevant descriptions were changed.	<a href="#">3.6.3</a>

In addition to the above changes, minor editorial corrections were made.



## Preface

This manual describes how to develop service templates and plug-ins used for JP1/Automatic Operation. In this manual, JP1/Automatic Operation is abbreviated to *JP1/AO*.

For reference information on JP1/AO manuals and a glossary, see the *JP1/Automatic Operation Overview and System Design Guide*.

### ■ Intended readers

This manual is intended for:

- Users who create new service templates
- Users who edit service templates

Readers of this manual must have a basic understanding of JP1/AO.

### ■ Microsoft product name abbreviations

This manual uses the following abbreviations for Microsoft product names.

Abbreviation			Full name or meaning
Hyper-V			Microsoft(R) Hyper-V(R)
Internet Explorer			Windows(R) Internet Explorer(R)
Windows	Windows 7		Microsoft(R) Windows(R) 7 Enterprise
			Microsoft(R) Windows(R) 7 Professional
			Microsoft(R) Windows(R) 7 Ultimate
	Windows 8.1		Microsoft(R) Windows(R) 8.1 Enterprise
			Microsoft(R) Windows(R) 8.1 Pro
	Windows 10		Microsoft(R) Windows(R) 10 Enterprise
			Microsoft(R) Windows(R) 10 Pro
	Windows Server 2008 R2	Windows Server 2008 R2 Datacenter	Microsoft(R) Windows Server(R) 2008 R2 Datacenter
		Windows Server 2008 R2 Enterprise	Microsoft(R) Windows Server(R) 2008 R2 Enterprise
		Windows Server 2008 R2 Standard	Microsoft(R) Windows Server(R) 2008 R2 Standard
	Windows Server 2012	Windows Server 2012 Datacenter	Microsoft(R) Windows Server(R) 2012 Datacenter
		Windows Server 2012 Standard	Microsoft(R) Windows Server(R) 2012 Standard
	Windows Server 2012 R2	Windows Server 2012 R2 Datacenter	Microsoft(R) Windows Server(R) 2012 R2 Datacenter

Abbreviation			Full name or meaning
Windows	Windows Server 2012 R2	Windows Server 2012 R2 Standard	Microsoft(R) Windows Server(R) 2012 R2 Standard
	Windows Server 2016	Windows Server 2016 Datacenter	Microsoft(R) Windows Server(R) 2016 Datacenter
		Windows Server 2016 Standard	Microsoft(R) Windows Server(R) 2016 Standard

## ■ Formatting conventions used in this manual

The following describes the formatting conventions used in this manual.

Text formatting	Description
<i>Character string</i>	Italic characters indicate a variable. Example: A date is specified in <i>YYYYMMDD</i> format.
<b>Bold - Bold</b>	Indicates selecting menu items in succession. Example: Select <b>File</b> - <b>New</b> . This example means that you select <b>New</b> from the <b>File</b> menu.
<b>key+key</b>	Indicates pressing keys on the keyboard at the same time. Example: <b>Ctrl+Alt+Delete</b> means pressing the <b>Ctrl</b> , <b>Alt</b> , and <b>Delete</b> keys at the same time.

## ■ Representation of JP1/AO-related installation folders

In this manual, the default installation folders for the Windows version of JP1/AO are represented as follows:

JP1/AO installation folder:

*system-drive*\Program Files\Hitachi\JP1AO

Common Component installation folder:

*system-drive*\Program Files\Hitachi\HiCommand\Base64

The installation folders for the Linux version of JP1/AO are as follows:

JP1/AO installation folder:

- /opt/jp1ao
- /var/opt/jp1ao

Common Component installation folder:

/opt/HiCommand/Base64

## ■ Diagrams of windows in the manual

Some windows in this manual might differ from the windows of your product because of reasons such as product improvements made without prior notice.

# Contents

Notices 2

Summary of amendments 4

Preface 5

## 1 Flow of Service Template Development 13

1.1 Overview 14

1.1.1 Flow of service template development 14

1.1.2 Elements involved in service template development 16

1.2 Main windows used to develop service templates 19

1.2.1 Transition of windows when developing service templates 19

1.2.2 **Service Builder** window 21

1.2.3 Procedure for starting editing of service templates 25

1.2.4 Notes when an interrupt operation is performed in the **Service Builder** window 26

1.3 Contents and structure of tasks associated with service templates 28

1.3.1 Contents and structure of tasks performed when creating a new service template 28

1.3.2 Contents and structure of tasks performed when editing and reusing a service template 29

1.3.3 Content and structure of tasks performed when using an existing service template as is 29

1.4 General procedure for creating new service templates 31

1.4.1 General procedure for creating new service templates 31

1.5 General procedure when editing and reusing an existing service template 33

1.5.1 General procedure when editing service template definition information 34

1.5.2 General procedure for editing a plug-in and applying the result to a service template 35

1.5.3 General procedure for creating new plug-ins and adding them to service templates 36

1.5.4 General procedure for changing the version of a component used as a step 37

1.5.5 General procedure for adding or deleting processing to or from a service template 38

1.5.6 General procedure for dynamically or statically setting property values when executing services 39

1.5.7 General procedure for setting service properties 41

1.6 Using existing service templates provided by JP1/AO 42

1.6.1 General procedure for using an existing service template provided by JP1/AO 42

1.7 List of service template development features 43

## 2 Setting Service Template Definition Information 45

2.1 Overview of development service templates and release service templates 46

2.2 Creating and changing the service template definition information 47

2.2.1 **Service Builder Edit** window **General** tab 47

2.2.2 Procedure for creating blank service templates 47

2.2.3 Procedure for changing the service template definition information 48

2.2.4	Items to set in service template definition information	49
2.2.5	Overview of custom files to be set to service templates	51
2.2.6	Procedure for setting custom files for service templates	52
2.2.7	Switching custom files for individual locales of the Web browser	53
2.2.8	Format of custom files	54
2.3	Setting display information for service templates in resource files	56
2.3.1	Procedure for setting service resource files	56
2.3.2	Format of service resource file	57
2.3.3	Definitions in service resource files	58
2.3.4	Correspondence between information displayed in service templates and properties in service resource files	59
2.3.5	Service resource files automatically generated when a service template is created	60
2.3.6	Service resource files updated when a service template is edited	61
2.3.7	Displaying a service template in a Web browser that is set to a locale for which no service resource file is available	62

## 3 Creating and Editing Flows for Service Templates 63

3.1	<b>Service Builder Edit</b> window <b>Flow</b> tab	64
3.2	Relationship between flow and steps	66
3.3	Creating flow hierarchies	67
3.4	Adding and editing steps	69
3.4.1	Procedure for adding steps	69
3.4.2	Procedure for editing steps	71
3.4.3	Settings in step definition information	72
3.4.4	Overview of subsequent step conditions	72
3.4.5	Conditional expressions that use arrows to indicate a connection with subsequent steps	73
3.5	Defining the execution order of steps	75
3.5.1	Procedure for defining the execution order of steps	75
3.5.2	Auto-completion of property values	76
3.5.3	Operations that can be performed on steps and relational lines	77
3.5.4	Information inherited when pasting steps or relational lines	77
3.5.5	Behavior when relational lines connect to multiple steps	78
3.5.6	Scenarios where relational lines cannot be drawn	78
3.5.7	Drawing relational lines when processing branches	79
3.6	Setting step properties	81
3.6.1	Overview of step properties	81
3.6.2	Procedure for directly specifying the input property values of steps	81
3.6.3	Procedure for mapping step property values	83
3.6.4	Overview of property mapping	85
3.6.5	Whether properties can be mapped depending on the visibility or data type	86
3.6.6	Procedure for elevating step properties to service properties	89
3.6.7	Example of defining step properties	90

3.6.8	List of reserved properties	92
3.6.9	Warning icon displayed for steps	95
3.7	Managing the versions of components used as steps	97
3.7.1	Overview of managing the versions of components used as steps	97
3.7.2	Procedure for checking the versions of components used as steps	98
3.7.3	Procedure for batch-updating components used as steps to the latest versions	98
3.7.4	Procedure for changing the version of a component used as a step to any specified version	100
3.7.5	Information inherited when the versions of components are changed	102

## **4      Setting Service Properties    104**

4.1	<b>Property</b> tab of the <b>Service Builder Edit</b> window	105
4.2	Editing and adding service properties	107
4.2.1	Overview of service property	107
4.2.2	Procedure for editing service properties	107
4.2.3	Procedure for adding service properties	108
4.2.4	Items set for input properties of services	109
4.2.5	Items set for output properties of services	111
4.2.6	Items set for variables	112
4.2.7	Visibility and display settings for properties	113
4.2.8	Procedure for deleting service properties	115
4.3	Service share properties	116
4.3.1	Overview of service share properties	116
4.3.2	Procedure for adding service share properties	117
4.3.3	Notes on defining service share properties	117
4.3.4	Overview of shared built-in service properties	118
4.4	Setting property groups	121
4.4.1	Procedure for setting property groups	121
4.4.2	Procedure for deleting property groups	122

## **5      Managing Service Templates    123**

5.1	Viewing service templates	124
5.1.1	Procedure for viewing service templates	124
5.2	Copying service templates	125
5.2.1	Procedure for copying service templates	125
5.2.2	Uniqueness of service templates and plug-ins	126
5.3	Deleting development service templates	128
5.3.1	Procedure for deleting development service templates	128
5.4	Releasing service templates	129
5.4.1	Overview of service template release	129
5.4.2	Procedure for releasing a service template	130
5.5	Exporting service templates	132
5.5.1	Procedure for exporting service templates	132

5.6	Importing service templates	133
5.6.1	Procedure for importing service templates	133
5.6.2	Importing service templates that contain steps using service components	134
5.6.3	Reason for maintaining separate development and active environments	134
<b>6</b>	<b>Creating and editing plug-ins</b>	<b>136</b>
6.1	Overview of plug-ins	137
6.1.1	Available operations by plug-in type	138
6.1.2	Plug-in executors	138
6.1.3	Files transferred to Windows systems	140
6.1.4	Files transferred to UNIX systems	141
6.1.5	Commands required for plug-in execution	141
6.1.6	Locale set for operation target devices during plug-in execution	141
6.1.7	Character set used for communication by JP1/AO during plug-in execution	142
6.1.8	Setting a specific character set during plug-in execution	143
6.2	Creating and editing plug-in definition information	145
6.2.1	Procedure for creating plug-ins	145
6.2.2	Procedure for editing plug-in definition information	147
6.2.3	Items to set in plug-in definition information	149
6.2.4	Image files that can be set for component icons	150
6.2.5	Plug-in credential types	151
6.3	Setting plug-in properties	152
6.3.1	Overview of plug-in properties	152
6.3.2	Procedure for adding plug-in properties	152
6.3.3	Procedure for editing plug-in properties	153
6.3.4	Items to set for plug-in input properties	154
6.3.5	Items to set for plug-in output properties	155
6.3.6	Reserved plug-in properties for specifying execution-target hosts and authentication information	155
6.3.7	Procedure for deleting plug-in properties	156
6.4	Editing platforms	158
6.4.1	Procedure for editing platforms	158
6.4.2	Items to set for platforms	159
6.4.3	Procedure for setting commands	160
6.4.4	Method for specifying scripts	161
6.4.5	Procedure for setting scripts (when attaching created scripts)	163
6.4.6	Procedure for setting scripts (when directly entering scripts)	164
6.4.7	Specifying commands in the <b>CLI Command</b> text box	165
6.4.8	Procedure for using the return value of a command or script as a flow branching condition (for values outside the 0 to 63 range)	167
6.4.9	Return values of content plug-ins	167
6.4.10	Relationship of command and script return values to the return values of plug-ins and steps	168
6.4.11	Information output to standard output by plug-ins	169

- 6.4.12 Procedure for mapping standard output and standard error output to output properties 169
- 6.4.13 Specifying **Output Filter** 171
- 6.4.14 Specifying **Execution Directory** 171
- 6.4.15 Procedure for adding and editing environment variables 172
- 6.4.16 Procedure for deleting environment variables 173
- 6.5 Using resource files to set plug-in display information 174
- 6.5.1 Procedure for setting plug-in resource files 174
- 6.5.2 Format of plug-in resource files 175
- 6.5.3 Correspondence between properties in plug-in resource files and information displayed for plug-ins 176
- 6.5.4 Plug-in resource files automatically generated when plug-ins are created 176
- 6.5.5 Plug-in resource files updated when plug-ins are edited 177
- 6.5.6 Displaying plug-ins by using a Web browser with a locale for which the plug-in resource file has not been created 177

## **7 Managing plug-ins 178**

- 7.1 Copying plug-ins 179
- 7.1.1 Procedure for copying plug-ins 179
- 7.2 Deleting plug-ins 181
- 7.2.1 Procedure for deleting plug-ins 181

## **8 Validating Service Templates 182**

- 8.1 Overview of service template validation 183
- 8.1.1 General procedure for validating service templates 183
- 8.1.2 Overview of building 184
- 8.1.3 Overview of debugging 185
- 8.1.4 Overview of operation tests 186
- 8.2 Building service templates 188
- 8.2.1 Procedure for building service templates 188
- 8.3 Debugging service templates 190
- 8.3.1 **Service Builder Debug** window 190
- 8.3.2 General procedure for debugging service templates 192
- 8.3.3 Functions used during debug operations 192
- 8.3.4 Example of debugging service templates 194
- 8.3.5 Procedure for starting debugging 195
- 8.3.6 Settings used when starting debugging 197
- 8.3.7 Procedure for debugging without pausing between steps 198
- 8.3.8 Timing with which step execution can be interrupted 198
- 8.3.9 Operations for interrupting step executions during debugging 199
- 8.3.10 Step information that can be changed when step execution is interrupted 202
- 8.3.11 Procedure for skipping plug-in processing during debugging 203
- 8.3.12 Plug-ins that cannot be interrupted or skipped during debugging 204

8.3.13	Procedure for checking property mapping settings during debugging	205
8.3.14	Procedure for changing step property values or return value during debugging	207
8.3.15	Importing and exporting step properties in the <b>Service Builder Debug</b> window	208
8.3.16	Displaying step property values and return values during debugging	208
8.3.17	Effect of changing values of step properties during debugging	209
8.3.18	Procedure for handling debug tasks that are waiting for a response (response entry)	210
8.3.19	Procedure for debugging a service template again without rebuilding	210
8.3.20	Procedure for retrying a task from a failed step during debugging	211
8.3.21	Procedure for retrying a task from the step after the failed step during debugging	212
8.3.22	Displaying debug task flow	212
8.3.23	Displaying the flow tree of a debug task	214
8.3.24	Displaying a repeated execution flow during debugging	215
8.3.25	Information displayed for repeated execution plug-ins and repeated execution flows during debugging	217
8.4	Managing debug tasks	218
8.4.1	Procedure for checking progress of debug tasks from the <b>Tasks</b> window	218
8.4.2	Procedure for checking details about debug tasks from the <b>Task Details</b> window	219
8.4.3	Procedure for checking task log entries for debug tasks	219
8.4.4	Procedure for stopping debug tasks	220
8.4.5	Procedure for forcibly stopping debug tasks	221
8.4.6	Procedure for deleting debug tasks	222
8.5	Testing the operation of service templates	223
8.5.1	Procedure for testing the operation of service templates	223

## Appendix 224

A	Reference Information	225
A.1	Reference information for build and release operations	225
A.2	Compatibility for service templates	229
A.3	Version changes	230

## Index 238



# 1

## Flow of Service Template Development

This chapter provides the general flow of service template development. Service templates are used to define processing that automates the operating procedures in an IT system.

## 1.1 Overview

---

This section describes the flow of service template development, and the elements involved in service template development.

### 1.1.1 Flow of service template development

In JP1/AO, you can use service templates to automate operating procedures. This functionality is particularly effective when applied to the automation of complex operating procedures, or procedures that are executed often but at irregular times.

You can create new service templates. You can also use the existing templates provided by JP1/AO<sup>#1</sup> without modification, or copy an existing template and edit it<sup>#2</sup> by adding and removing steps as needed.

#1

Service templates provided by JP1/AO include the service templates provided with the JP1/AO standard package and the JP1/AO Content Set (available separately).

#2

Before you can edit a service template provided by JP1/AO, you need to import the service template.



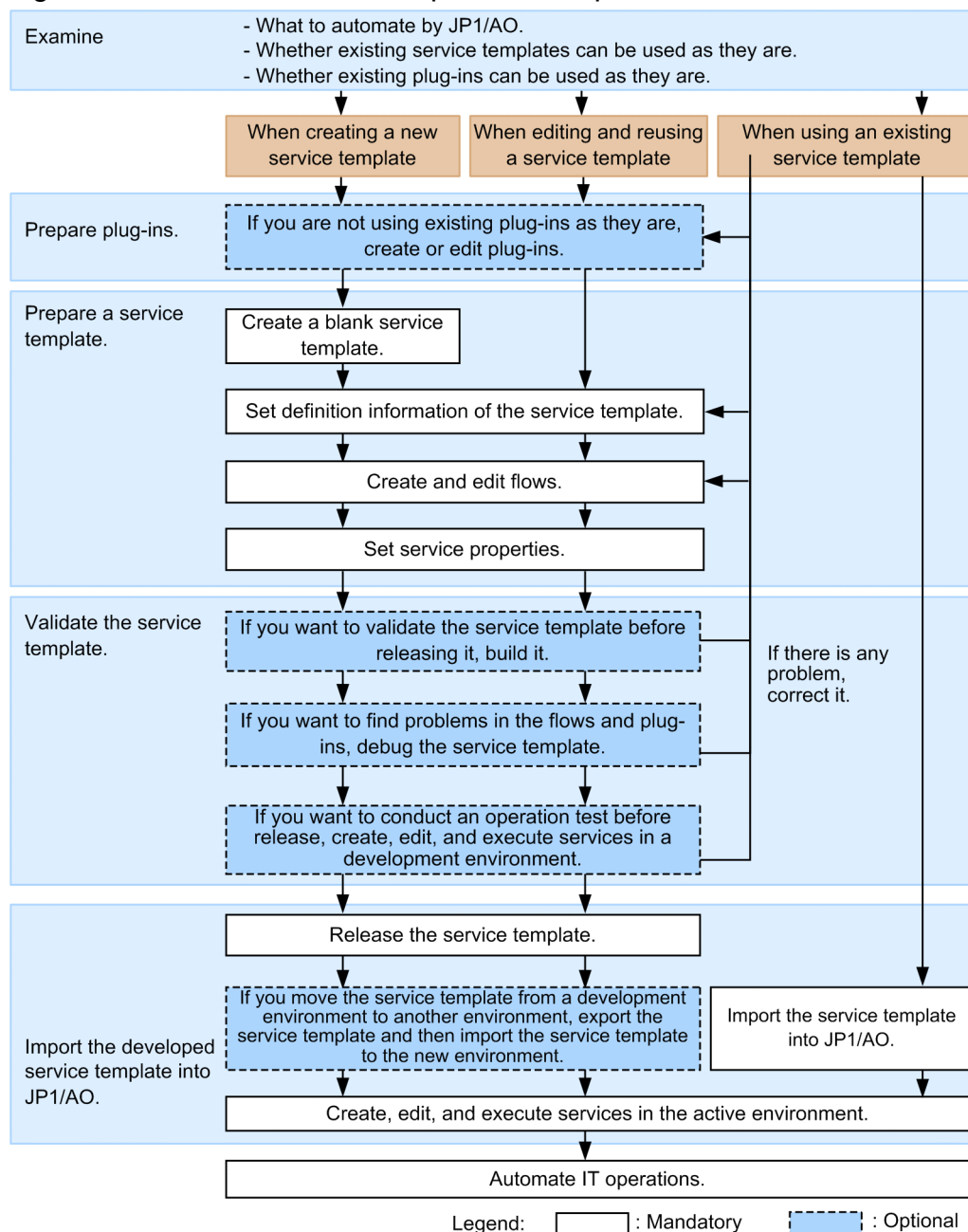
#### Tip

If you want to use a content plug-in provided by JP1/AO as the basis for service template development, import the service template that contains the content plug-in. *Type* Components (such as "HCS Components" and "Hyper-V2008 Components") are provided as the service templates for providing plug-ins.

For details about the plug-ins contained in these service templates, see *List of plug-ins contained in service templates* in the manual *JP1/Automatic Operation Service Template Reference*.

The following figure shows the general procedure for developing service templates.

Figure 1-1: Flow of service template development



### Tip

Detailed input restrictions are not set for the properties of the plug-ins provided by JP1/AO because they were created for general-purpose use. If necessary, when using these plug-ins to create a service template, consider adding input restrictions on service properties (definition information of the service).

## Related topics for creating new service templates

- [1.1.2 Elements involved in service template development](#)
- [1.2 Main windows used to develop service templates](#)
- [1.3.1 Contents and structure of tasks performed when creating a new service template](#)
- [1.4 General procedure for creating new service templates](#)

## **Related topics for editing service templates**

- [1.1.2 Elements involved in service template development](#)
- [1.2 Main windows used to develop service templates](#)
- [1.3.2 Contents and structure of tasks performed when editing and reusing a service template](#)
- [1.5 General procedure when editing and reusing an existing service template](#)

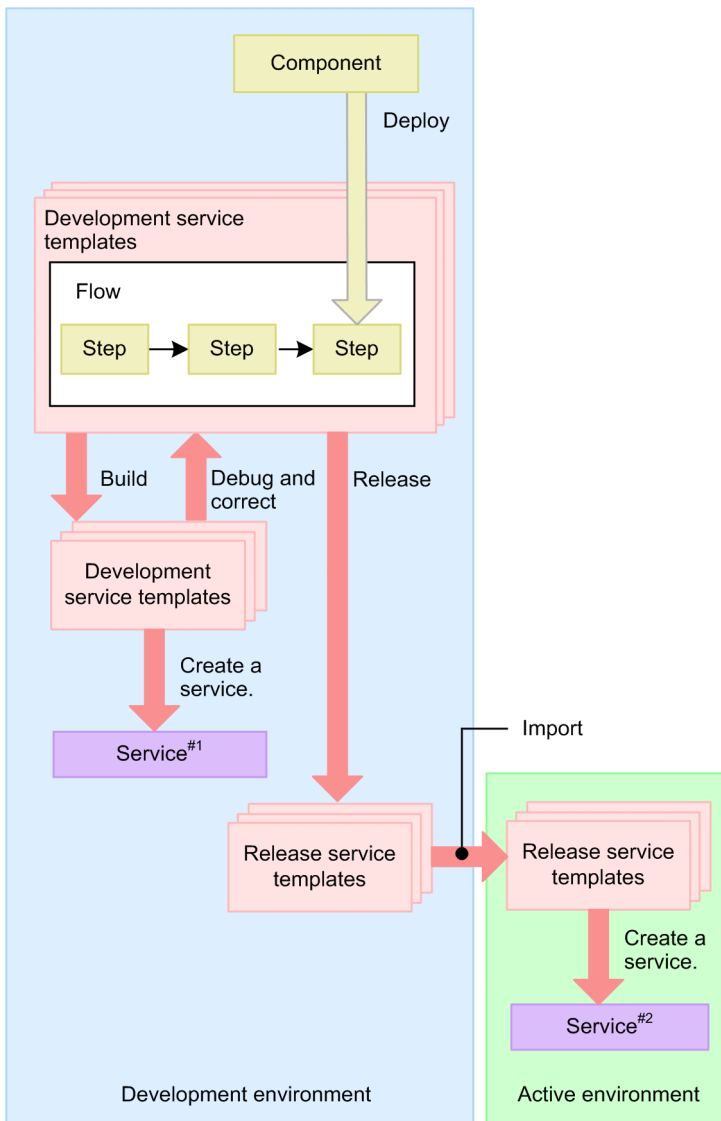
## **Related topics for using unmodified service templates**

- [1.1.2 Elements involved in service template development](#)
- [1.3.3 Content and structure of tasks performed when using an existing service template as is](#)
- [1.6 Using existing service templates provided by JP1/AO](#)

### **1.1.2 Elements involved in service template development**

Service templates define the information necessary to automate operating procedures. The following figure shows the elements involved in service template development.

Figure 1-2: Elements Involved in service template development



- #1:  
A service used to test the service template in the development environment.
- #2:  
A service to be actually executed in the active environment.

- **Development environment**  
The environment in which the service template is developed. You can also conduct debugging and operation testing in this environment to validate the operation of the service templates you develop. Although you can develop service templates in an active environment, we recommend that you keep the development and active environments separate.
- **Active environment**  
The environment in which you can create and execute services based on service templates you have developed. The actual automation of operating procedures takes place in this environment.
- **Service template**  
Defines the processing that automates the operating procedures in an IT system. A service template incorporates flows and steps.
- **Development service template**

A service template that is being developed by a user. Service templates created by copying a release service template are also classified as development service templates. Development service templates are used in the development environment.

- Release service template

When you release a development service template, it becomes a release service template, which cannot be edited. The service templates provided by JP1/AO are also classified as release service templates. Release service templates are used in the active environment.

- Flow

Defines the flow of the operating procedure you are automating.

- Component

Plug-ins and release service templates that can be placed as steps.

- Plug-in

The smallest unit of processing you can define when automating IT operations.

- Step

An element of a flow. Each step executes a plug-in.

## 1.2 Main windows used to develop service templates

---

The **Service Builder** window is mainly used to develop service templates. Some operations can also be performed in the **Service Template** window.

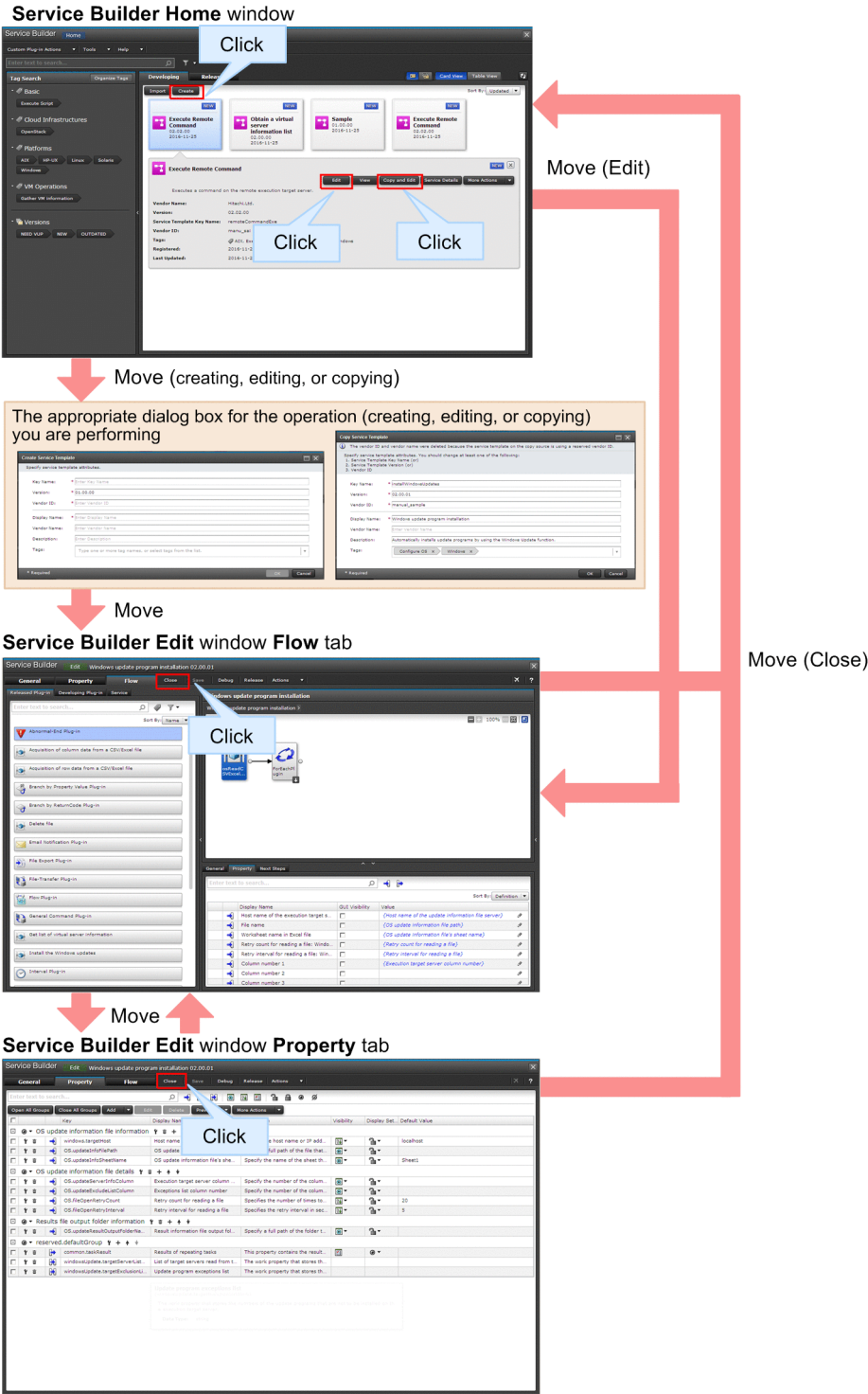
### 1.2.1 Transition of windows when developing service templates

The display of the **Service Builder** window can be switched between the **Service Builder Home** page and the **Service Builder Edit** page.

The **Service Builder Home** window is the base window used to start operations regarding service template development. When you develop service templates, first come to this window and select an operation to perform.

To edit a service template, in the **Service Builder Edit** window, first edit the template in the **Flow** tab and then configure the service properties in the **Property** tab. Note that you can switch between the **Flow** and **Property** tabs of the **Service Builder Edit** window at any time while editing a service template.

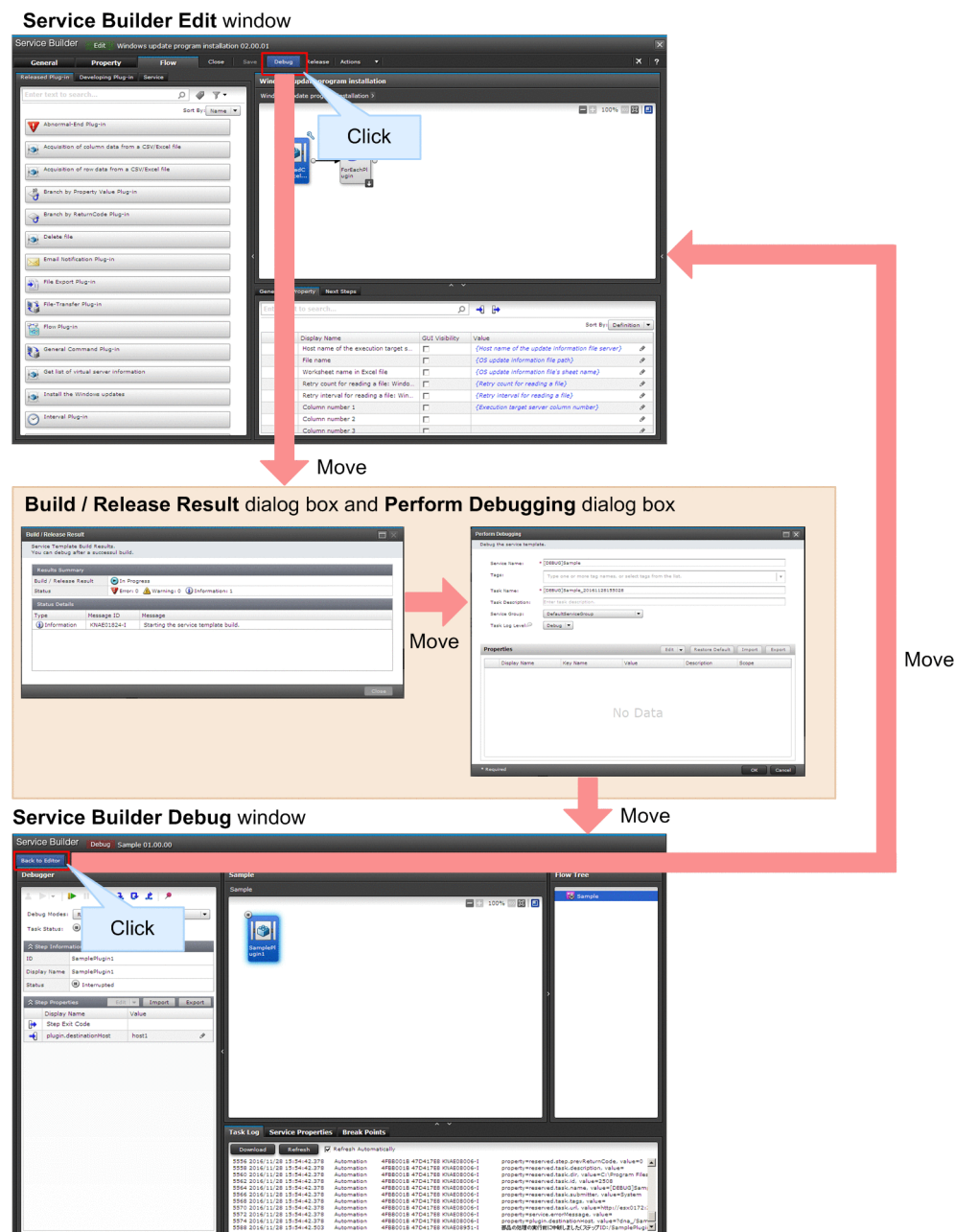
Figure 1-3: Main windows used in service template development (when creating, editing, or copying a service template)



After editing a service template, in the **Service Builder Debug** window, debug the service template.



Figure 1-4: Main windows used in service template development (when debugging a service template)



## Related topics

- [1.2.2 Service Builder window](#)
- [8.3.1 Service Builder Debug window](#)

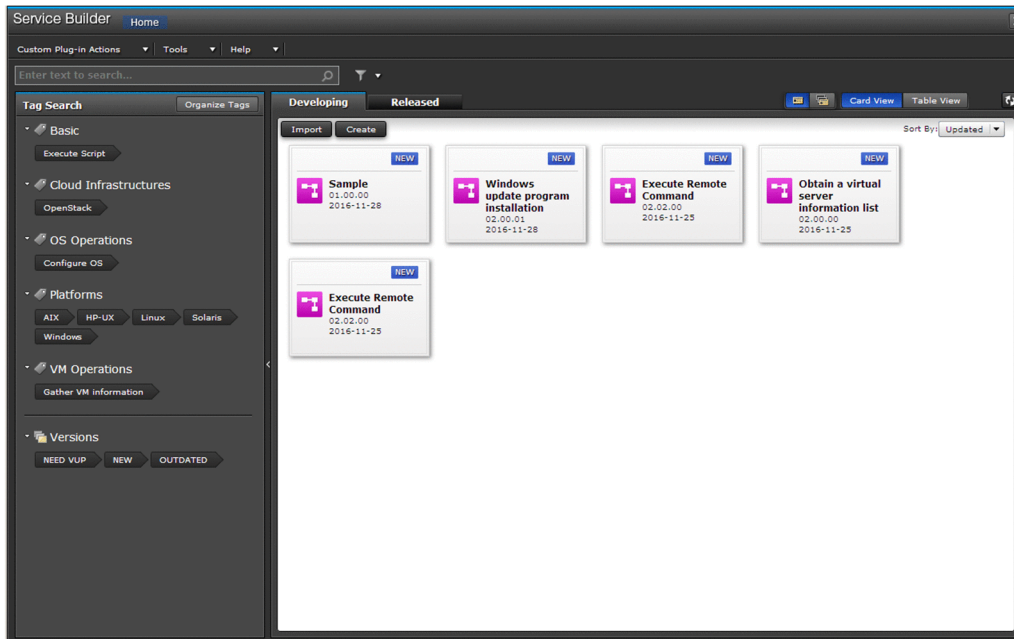
## 1.2.2 Service Builder window

The **Service Builder** window is used to develop service templates and plug-ins. To display the **Service Builder** window, in the main window, select **Tools** and then **Service Builder**.

## Service Builder Home window

The **Service Builder Home** window is used as the base window to start development of service templates and plug-ins.

Figure 1-5: **Service Builder Home** window



The **Service Builder Home** window displays service templates that were created (in the **Released** tab) and service templates that are currently being developed (in the **Developing** tab). You can check details about a selected service template or use the management functions to edit, view, copy, export, or delete service templates. You can also import and create new service templates as needed.

Use the text box in the upper left corner of the window to search for a specific service template. You can also search for a service template by the tag groups associated to that service template. Use the **Card View** tab or the **Table View** tab to change the way service templates are displayed.

The following describes the items displayed in the window:

### Custom Plug-in Actions pull-down menu

Specify the operation to be performed on a plug-in:

**Create:** Create a new development plug-ins.

**Edit:** Edit a development plug-ins.

**Set Resources:** Specify a resource file for a development plug-ins.

**Copy:** Copy a development plug-ins or a release plug-ins.

**Delete:** Delete a development plug-ins or a release plug-ins.

### Tools pull-down menu

**Reset Preferences** returns the settings in the window back to their initial values at the time of shipment.

### Help pull-down menu

**Online Manual** displays the JP1/AO manual.

### Close button

In the dialog box asking whether to close the application, clicking the **OK** button closes the **Service Builder Home** window and returns you to the main window.

## Developing tab

### **Import** button

Import a development service template or a release service template into the JP1/AO server.

### **Create New Service Template** button

Create a new development service template.

### **Edit** button

Edit a selected development service template.

### **View** button

View a selected development service template.

### **Copy** button

Copy a selected development service template.

### **Service Details** button

Check details about a selected service template.

### **More Actions** pull-down menu

**Export:** Export the content of a selected service template to a file you specify.

**Delete:** Delete a selected service template.

## Released tab

### **View** button

View a release service template you select.

### **Copy and Edit** button

Copy a release service template you select.

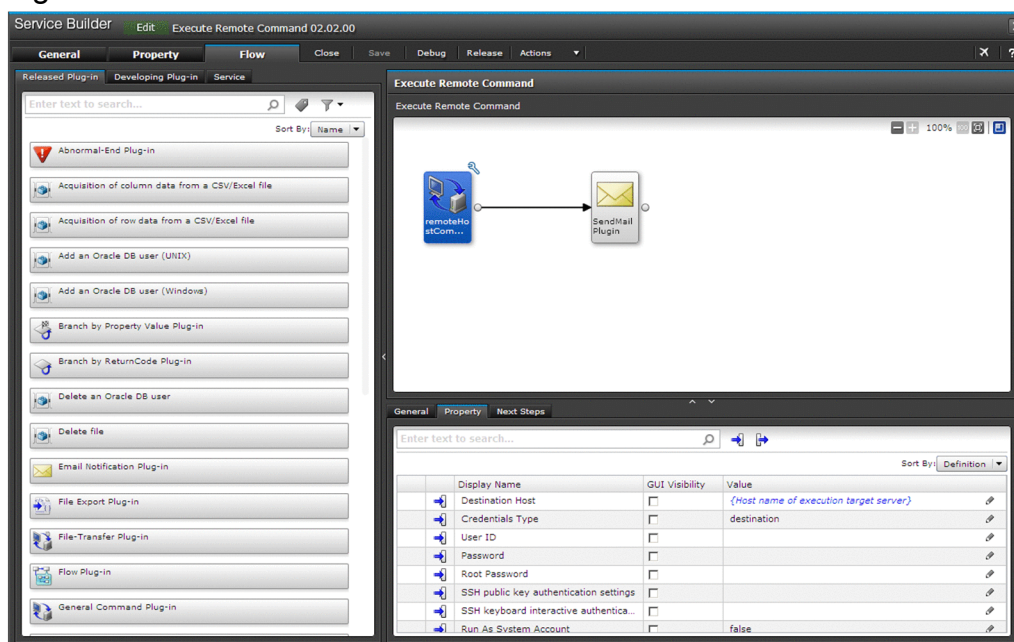
### **Service Details** button

Check the details about a release service template you select.

## Service Builder Edit window

The **Service Builder Edit** window is used to edit service template flows and properties.

Figure 1-6: **Service Builder Edit** window



You can edit service templates in the **Service Builder Edit** window. Switch between tabs to view or edit details about a service template, the general settings, and the properties.

The following describes the items displayed in the window:

#### Flow tab

Create or edit a service template flow. For details, see the topic [3.1 Service Builder Edit window Flow tab](#).

#### General tab

Check the details about a service template. Click the **Edit** button to edit or customize a service template. For details, see the topic [2.2.1 Service Builder Edit window General tab](#).

#### Property tab

View the input and output properties related to a service template. For details, see the topic [4.1 Property tab of the Service Builder Edit window](#).

#### Close button

Click this button to return to the **Service Builder Home** window.

#### Save button

Click this button to save a service template.

#### Debug button

Click this button to build a service template. If the build is successful, you will be able to debug the service template.

#### Release button

Click this button to release a service template.

#### Actions pull-down menu

Select the **Component Version Management** button to manage the versions of the components that are used as steps of the service template.

---

## Related topics

- [1.2.1 Transition of windows when developing service templates](#)
-

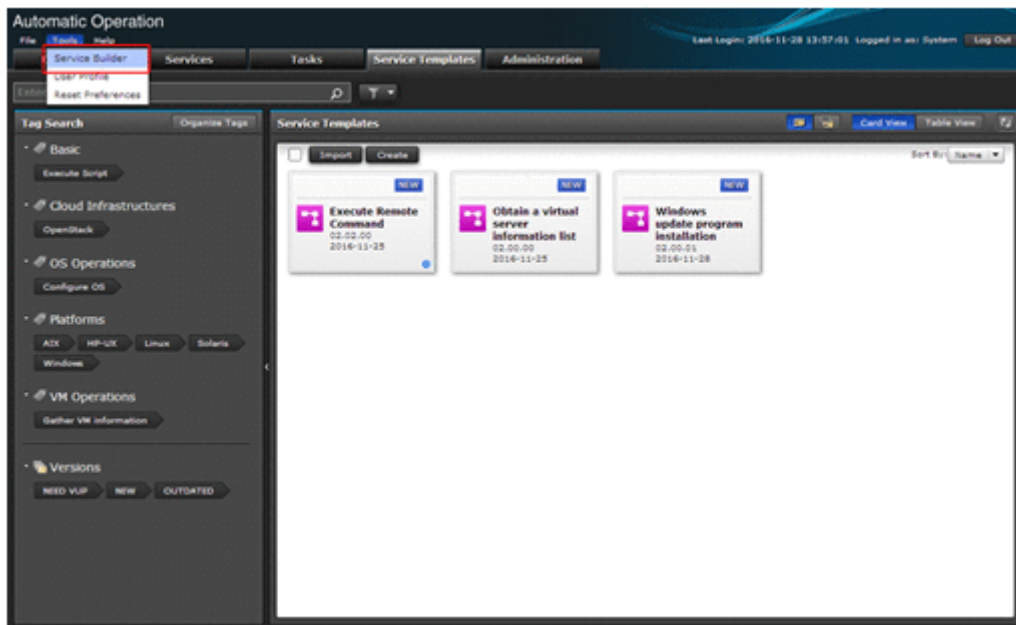
## 1.2.3 Procedure for starting editing of service templates

You can edit service templates in the **Service Builder Edit** window. Note that, to edit a release service template, you must first create a copy of the release service template as a development service template and then edit the development service template.

### To start editing of service template:

1. In the main window, from the **Tools** menu, select **Service Builder**.

Figure 1-7: Selecting **Service Builder**



2. Select the development service template that you want to edit, and then click the **Edit** button.

### Operation result

The **Flow** tab of the **Service Builder Edit** window appears. Create or edit a flow, or configure the service properties as needed.



#### Tip

To edit a release service template, click the **Copy and Edit** button on the **Service Builder Home > Released** tab. After you set the general settings (in the **Copy Service Template** dialog box), the **Flow** tab of the **Service Builder Edit** window automatically appears.

### Related topics

- [2.1 Overview of development service templates and release service templates](#)
- [5.2.1 Procedure for copying service templates](#)
- [3.4 Adding and editing steps](#)
- [4.2 Editing and adding service properties](#)

## 1.2.4 Notes when an interrupt operation is performed in the Service Builder window

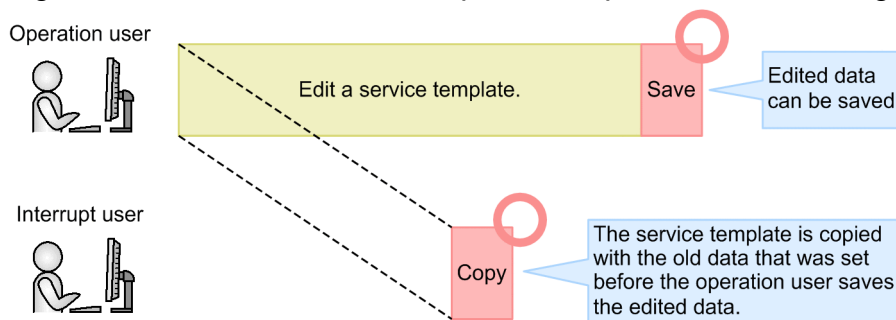
This subsection describes the behavior when another user performs an interrupt operation on a service template or plug-in while you are editing the same service template or plug-in. The examples for service templates are shown below. You can expect the same behavior for plug-ins.

In these examples, the user who is editing a service template in the **Service Builder** window is called the *operation user*. Also, the user who performs different operations on the service template that the operation user is operating, is called the *interrupt user*.

- When a service template is copied while it is being edited

Even while the operation user is editing a service template, the interrupt user can copy the same service template. The content of the service template created by the copy operation is the same as the content before the operation user saves the editing result of the service template.

Figure 1-8: When a service template is copied while it is being edited

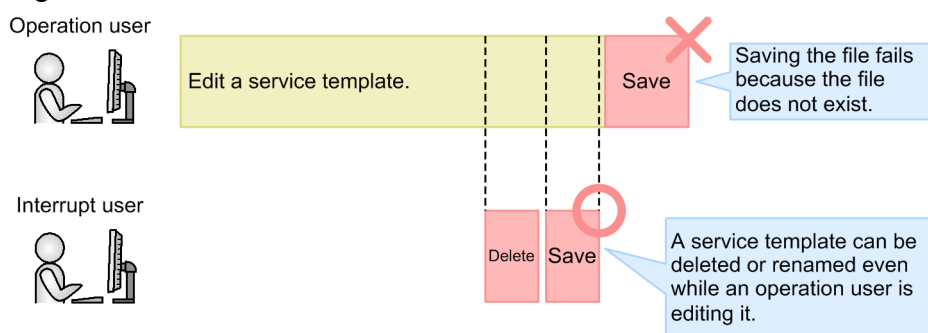


- When a file is deleted or renamed while a service template or plug-in is being edited

While the operation user is editing a service template or plug-in, if the interrupt user performs one of the operations below on the same service template or plug-in, an error message appears. This error message appears when the operation user saves the service template or plug-in, or when the operation user builds or releases the service template.

- The plug-in icon set for the plug-in is deleted (except for the standard plug-in icon).
- The script file set for the plug-in is deleted or renamed.
- The custom file set for the service template is deleted or renamed.

Figure 1-9: When a file is deleted or renamed while a service template is being edited

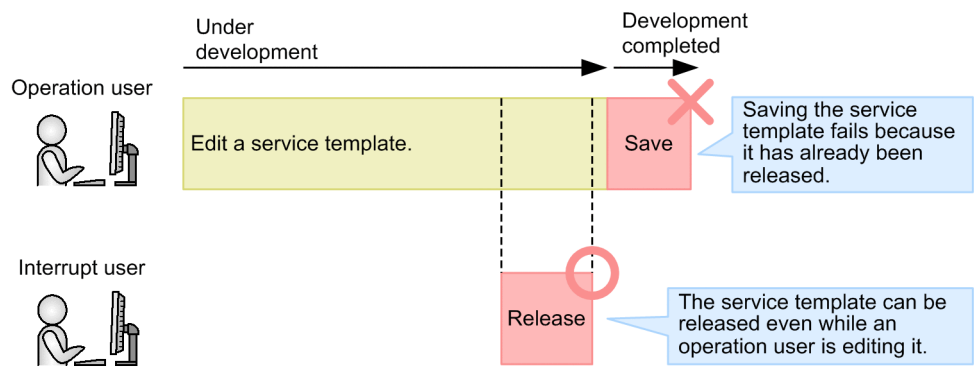


- When a service template is released while it is being edited

While the operation user is editing a service template, if the interrupt user releases the same service template, the release processing is performed successfully. However, the result of the release processing is not applied to the window in which the operation user is editing the service template. When the operation user saves the editing result of the service template, the save processing fails, and an error message appears.

When you release a service template, you need to confirm that no other users are editing the same service template.

Figure 1-10: When a service template is released while it is being edited





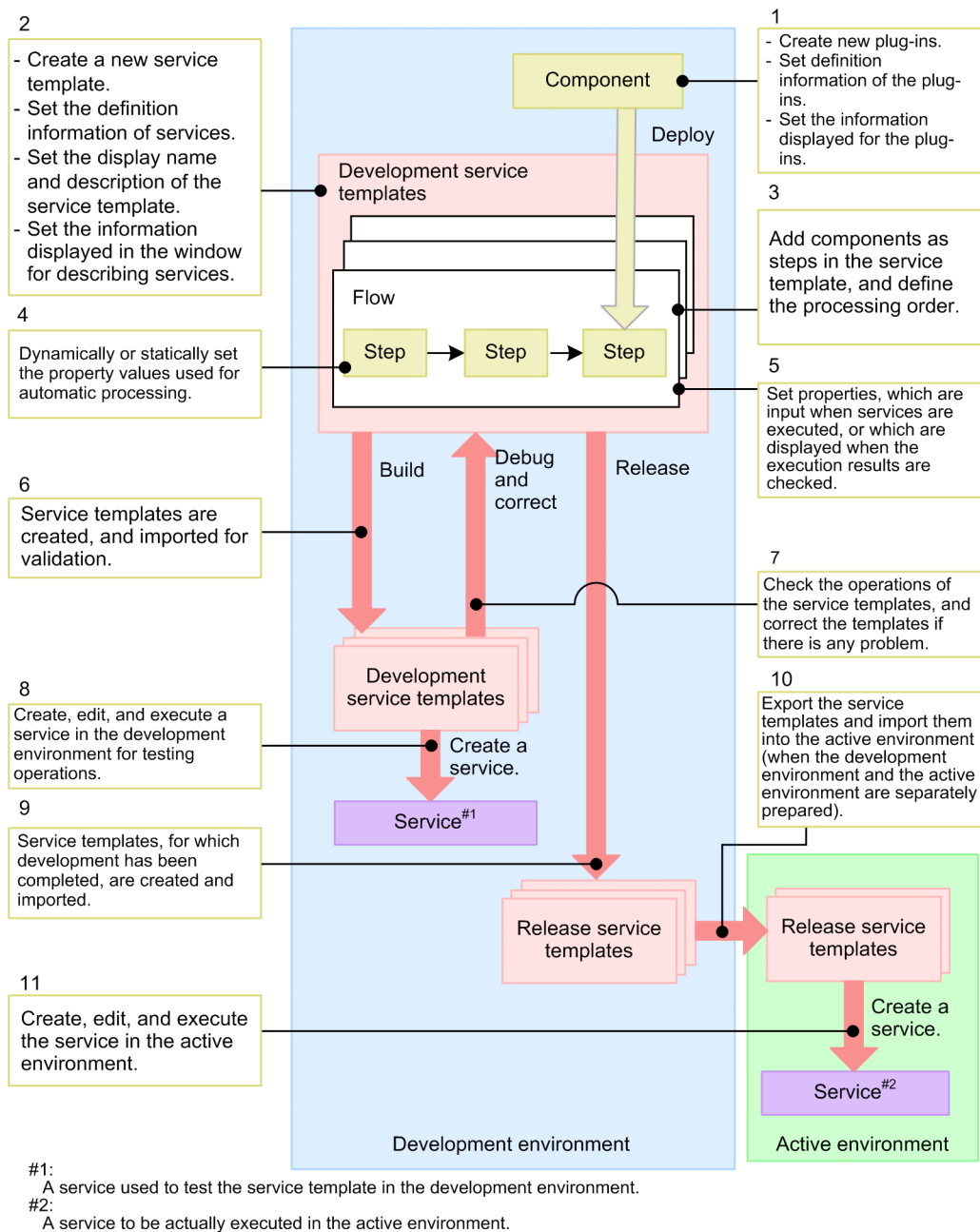
## 1.3 Contents and structure of tasks associated with service templates

This section describes the contents and structure of tasks performed when you develop service templates or use the service templates provided by JP1/AO.

### 1.3.1 Contents and structure of tasks performed when creating a new service template

The figure below shows the contents and structure of tasks performed when creating a new service template. The numbers in the figure indicate the order in which each step is performed.

Figure 1-11: Contents and structure of tasks performed when creating a new service template

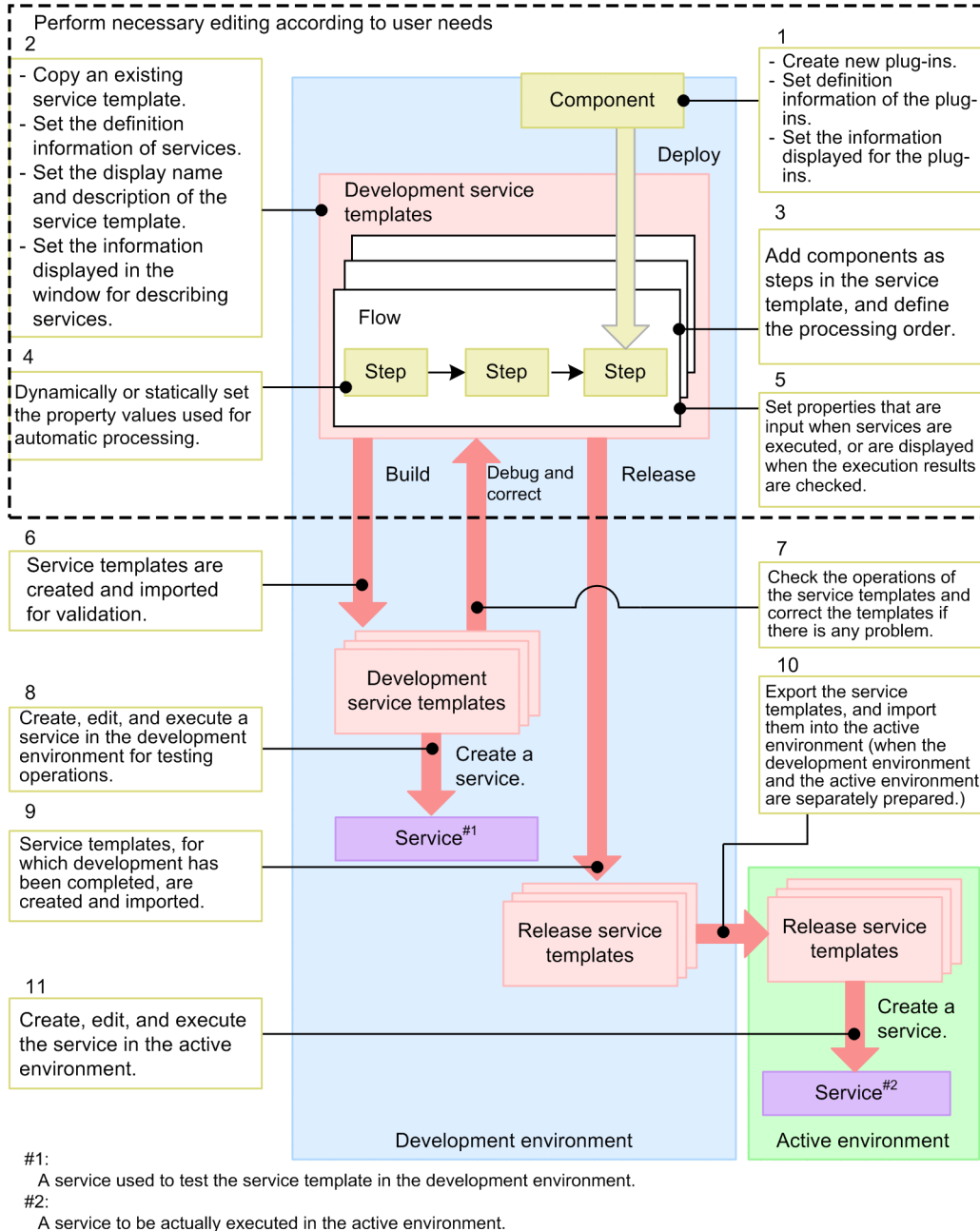




## 1.3.2 Contents and structure of tasks performed when editing and reusing a service template

The figure below shows the contents and structure of tasks performed when editing and reusing a service template. The numbers in the figure indicate the order in which each step is performed. Perform the tasks in the section enclosed by the dashed line only if they are needed.

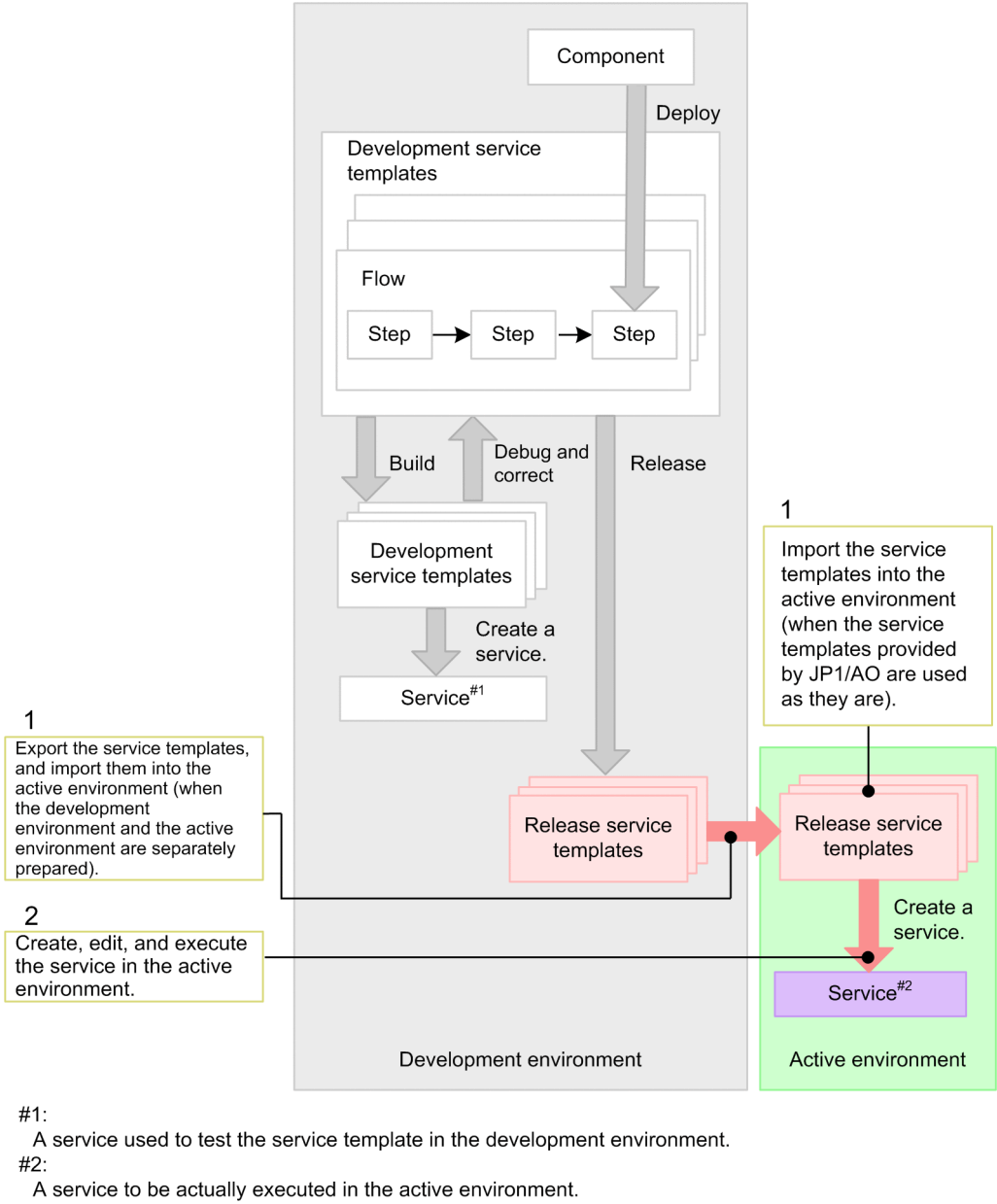
Figure 1-12: Contents and structure of tasks performed when editing and reusing a service template



## 1.3.3 Content and structure of tasks performed when using an existing service template as is

The figure below shows the content and structure of tasks performed when using an existing service template as is. The numbers in the figure indicate the order in which each step is performed.

Figure 1-13: Content and structure of tasks performed when using an existing service template as is



## 1.4 General procedure for creating new service templates

This section describes the general procedure for creating a new service template, and gives references to manuals.

### 1.4.1 General procedure for creating new service templates

Users are not limited to using and modifying the service templates provided by JP1/AO, and can create new service templates to meet specific needs.

#### You might use this procedure when:

- You want to create your own service template rather than use an existing one.

#### Required knowledge

- [2.1 Overview of development service templates and release service templates](#)
- [2.2.4 Items to set in service template definition information](#)
- [6.2.3 Items to set in plug-in definition information](#)
- [5.4.1 Overview of service template release](#)

#### General procedure

Table 1-1: General procedure for creating new service templates

Task		Mandatory/ optional	Refer to
1	Create a blank service template.	Mandatory	<a href="#">2.2.2 Procedure for creating blank service templates</a>
2	If not using an existing plug-in, create a new plug-in.	Optional	<a href="#">6.2.1 Procedure for creating plug-ins</a>
	To edit and reuse a release plug-in, copy the plug-in and edit the copy.	Optional	<a href="#">7.1.1 Procedure for copying plug-ins</a> <a href="#">6.2.2 Procedure for editing plug-in definition information</a>
	To edit and reuse a development plug-in, begin by editing the plug-in.	Optional	<a href="#">6.2.2 Procedure for editing plug-in definition information</a>
3	Create a flow.	Mandatory	<a href="#">3. Creating and Editing Flows for Service Templates</a>
4	Set service properties.	Mandatory	<a href="#">4. Setting Service Properties</a>
5	Validate the created service template. <sup>#</sup>	Optional	<a href="#">8. Validating Service Templates</a>
6	Release the completed service template and prepare to create the service.	Mandatory	<a href="#">5.4.2 Procedure for releasing a service template</a>
7	To move the service template from the development environment to another environment, export the service template,	Optional	<a href="#">5.5.1 Procedure for exporting service templates</a> , <a href="#">5.6.1 Procedure for importing service templates</a>

1. Flow of Service Template Development

Task		Mandatory/ optional	Refer to
7	and then import it to the destination environment.	Optional	<a href="#">5.5.1 Procedure for exporting service templates</a> , <a href="#">5.6.1 Procedure for importing service templates</a>
8	Create, edit, and execute the service.	Mandatory	<i>JP1/Automatic Operation Administration Guide</i> <ul style="list-style-type: none"> <li>• Creating services</li> <li>• Editing services</li> <li>• Executing services</li> </ul>

#

If the validation process finds any issues with the service template, repeat tasks 2 to 5 as needed.

## 1.5 General procedure when editing and reusing an existing service template

Users can create original service templates by copying and then editing the service templates provided by JP1/AO (in the standard JP1/AO package or JP1/AO Content Set). To edit a release service template, copy the template and then edit the copy as a new service template.

### Important

Once edited, the service templates (bundled service templates and JP1/AO Content Set) and plug-ins provided by JP1/AO are outside the scope of JP1/AO product support. However, the plug-ins provided by JP1/AO (the standard JP1/AO package or JP1/AO Content Set) that are called from such templates remain subject to product support.

The following table lists the cases of editing service templates and shows where to read about the tasks involved in editing a service template.

Table 1-2: Reading material for each development stage

Task	You might perform this task when:	Refer to
Investigation	<ul style="list-style-type: none"><li>You are investigating whether an existing service template or plug-in can be used without further modification.</li></ul>	<i>JP1/Automatic Operation Service Template Reference</i>
Editing service template definition information	<ul style="list-style-type: none"><li>You want to change the name of the service template from "Stop virtual server" to "Stop virtual server and Notify by email".</li></ul>	<a href="#">1.5.1 General procedure when editing service template definition information</a>
Editing plug-ins	<ul style="list-style-type: none"><li>You want to change the contents of scripts or commands defined in a plug-in.</li><li>You want to change the icon displayed for a plug-in in the <b>Flow</b> area.</li></ul>	<a href="#">1.5.2 General procedure for editing a plug-in and applying the result to a service template</a>
Creating new plug-ins	<ul style="list-style-type: none"><li>You want to create a new plug-in and define processing that executes a command.</li></ul>	<a href="#">1.5.3 General procedure for creating new plug-ins and adding them to service templates</a>
Changing the version of components used as steps	<ul style="list-style-type: none"><li>You want to replace the plug-ins used as the steps with the latest version of plug-ins.</li></ul>	<a href="#">1.5.4 General procedure for changing the version of a component used as a step</a>
Adding or deleting steps to or from the service template	<ul style="list-style-type: none"><li>You want to insert an email-sending process at the end of the processing automated by the service template.</li><li>A file transfer step is no longer required for processing that acquires log data for JP1/IM and JP1/Base.</li></ul>	<a href="#">1.5.5 General procedure for adding or deleting processing to or from a service template</a>
Setting property values dynamically or statically during execution of the service	<ul style="list-style-type: none"><li>For processing that increases available memory, you want the memory capacity to be fixed at 5 GB each time the service runs.</li></ul>	<a href="#">1.5.6 General procedure for dynamically or statically setting property values when executing services</a>

Task	You might perform this task when:	Refer to
Setting property values dynamically or statically during execution of the service	<ul style="list-style-type: none"> <li>You want a command to be executed in response to the execution result of the previous command.</li> </ul>	<a href="#">1.5.6 General procedure for dynamically or statically setting property values when executing services</a>
Setting properties used by users who execute the service to specify necessary values for executing the service, and to acquire the execution result of the service	<ul style="list-style-type: none"> <li>For processing that increases the memory capacity, you want to specify the memory capacity (instead of fixing the memory capacity) when executing the service.</li> <li>You want to check the processing result of a plug-in in the <b>Task Details</b> window.</li> </ul>	<a href="#">1.5.7 General procedure for setting service properties</a>

## 1.5.1 General procedure when editing service template definition information

Service template definition information is the name and description of the service template displayed in the JP1/AO operation window.

### You might use this procedure when:

- You want to change the name of the service template from "Stop virtual server" to "Stop virtual server and Notify by email".

### Required knowledge

- [2.1 Overview of development service templates and release service templates](#)
- [2.2.4 Items to set in service template definition information](#)
- [5.4.1 Overview of service template release](#)

### General procedure

Table 1-3: General procedure when editing service template definition information

Task		Man dator y/ optio nal	Refer to
1	Copy the release service template that you want to edit.	Optio nal	<a href="#">5.2.1 Procedure for copying service templates</a>
2	Change the service template definition information.	Mand atory	<a href="#">2.2.3 Procedure for changing the service template definition information</a>
3	Release the completed service template and prepare to create the service.	Mand atory	<a href="#">5.4.2 Procedure for releasing a service template</a>
4	To move the service template from the development environment to another environment, export the service template,	Optio nal	<a href="#">5.5.1 Procedure for exporting service templates</a> , <a href="#">5.6.1 Procedure for importing service templates</a>

Task		Mandatory/ optional	Refer to
4	and then import it to the destination environment.	Optional	<a href="#">5.5.1 Procedure for exporting service templates</a> , <a href="#">5.6.1 Procedure for importing service templates</a>
5	Create, edit, and execute the service.	Mandatory	<i>JP1/Automatic Operation Administration Guide</i> <ul style="list-style-type: none"> <li>• Creating services</li> <li>• Editing services</li> <li>• Executing services</li> </ul>

## 1.5.2 General procedure for editing a plug-in and applying the result to a service template

You can edit plug-ins (except for basic plug-ins). After editing a plug-in, you can apply the result to a service template. If the plug-in is already released, copy the plug-in and edit the copy.

### You might use this procedure when:

- You want to change the contents of a script or command defined in a plug-in.
- You want to change the icon displayed for a plug-in in the **Flow** area.

### Required knowledge

- [6.1 Overview of plug-ins](#)
- [2.1 Overview of development service templates and release service templates](#)
- [5.4.1 Overview of service template release](#)

### General procedure

Table 1-4: General procedure for editing a plug-in and applying the result to a service template

Task		Mandatory/ optional	Refer to
1	Copy the release plug-in that you want to edit.	Optional	<a href="#">7.1.1 Procedure for copying plug-ins</a>
2	Edit plug-in definition information.	Mandatory	<a href="#">6.2.2 Procedure for editing plug-in definition information</a>
3	Copy the service template when applying a plug-in to a release service template.	Optional	<a href="#">5.2.1 Procedure for copying service templates</a>
4	Add the edited plug-in as a step in a flow, or if the same version of the plug-in has already been used as a step, change the plug-in to a newer version.	Mandatory	<a href="#">3.4.1 Procedure for adding steps</a> , <a href="#">3.7.4 Procedure for changing the version of a component used as a step to any specified version</a>
5	Add or delete related steps when editing a plug-in affects the flow of processing.	Optional	<a href="#">3.4 Adding and editing steps</a> , <a href="#">3.5.3 Operations that can be performed on steps and relational lines</a>

Task		Mandatory/optional	Refer to
6	Set step properties when editing a plug-in affects step properties.	Optional	<a href="#">3.6 Setting step properties</a>
7	Set service properties when editing a plug-in affects service properties.	Optional	<a href="#">4.2 Editing and adding service properties</a>
8	Validate the edited service template <sup>#</sup> .	Optional	<a href="#">8. Validating Service Templates</a>
9	Release the completed service template and prepare to create the service.	Mandatory	<a href="#">5.4.2 Procedure for releasing a service template</a>
10	To move the service template from the development environment to another environment, export the service template, and then import it to the destination environment.	Optional	<a href="#">5.5.1 Procedure for exporting service templates</a> , <a href="#">5.6.1 Procedure for importing service templates</a>
11	Create, edit, and execute the service.	Mandatory	<i>JP1/Automatic Operation Administration Guide</i> <ul style="list-style-type: none"> <li>• Creating services</li> <li>• Editing services</li> <li>• Executing services</li> </ul>

#

If the validation process finds an issue with the service template, repeat tasks 2 to 8 as needed.

## 1.5.3 General procedure for creating new plug-ins and adding them to service templates

Users can create custom plug-ins and add them to service templates as steps.

### You might use this procedure when:

- You want to create a new plug-in and define processing that executes a command.

### Required knowledge

- [6.1 Overview of plug-ins](#)
- [2.1 Overview of development service templates and release service templates](#)
- [5.4.1 Overview of service template release](#)

### General procedure

Table 1-5: General procedure for creating new plug-ins and adding them to service templates

Task		Mandatory/optional	Refer to
1	Create a plug-in.	Mandatory	<a href="#">6.2.1 Procedure for creating plug-ins</a>



Task		Mandatory/optional	Refer to
2	Copy the release service template that you want to edit.	Optional	<a href="#">5.2.1 Procedure for copying service templates</a>
3	Add a plug-in you created as a step in a flow.	Mandatory	<a href="#">3.4.1 Procedure for adding steps</a>
4	Add and delete related steps when adding a step affects the flow of processing.	Optional	<a href="#">3.4 Adding and editing steps</a> , <a href="#">3.5.3 Operations that can be performed on steps and relational lines</a>
5	Set step properties when editing a plug-in affects step properties.	Optional	<a href="#">3.6 Setting step properties</a>
6	Set service properties when editing a plug-in affects service properties.	Optional	<a href="#">4.2 Editing and adding service properties</a>
7	Validate the edited service template <sup>#</sup> .	Optional	<a href="#">8. Validating Service Templates</a>
8	Release a completed service template and prepare to create the service.	Mandatory	<a href="#">5.4.2 Procedure for releasing a service template</a>
9	To move the service template from the development environment to another environment, export the service template, and then import it to the destination environment.	Optional	<a href="#">5.5.1 Procedure for exporting service templates</a> , <a href="#">5.6.1 Procedure for importing service templates</a>
10	Create, edit, and execute the service.	Mandatory	<i>JP1/Automatic Operation Administration Guide</i> <ul style="list-style-type: none"> <li>• <a href="#">Creating services</a></li> <li>• <a href="#">Editing services</a></li> <li>• <a href="#">Executing services</a></li> </ul>

#

If the validation process finds an issue with the service template, repeat tasks 1 to 7 as needed.

## 1.5.4 General procedure for changing the version of a component used as a step

You can replace a component used as a step in a service template, with another version of the component.

### You might use this procedure when:

- You want to replace a plug-in used as a step, with the latest version of the plug-in.

### Required knowledge

- [6.1 Overview of plug-ins](#)
- [2.1 Overview of development service templates and release service templates](#)
- [3.7.1 Overview of managing the versions of components used as steps](#)
- [5.4.1 Overview of service template release](#)

## General procedure

Table 1-6: General procedure for changing the version of a component used as a step

Task		Mandatory/ optional	Refer to
1	Copy the service template when applying a plug-in to a release service template.	Optional	<a href="#">5.2.1 Procedure for copying service templates</a>
2	Check the version of the component used by a step, and change the version as needed.	Mandatory	<a href="#">3.7 Managing the versions of components used as steps</a>
3	Set step properties when changing the version of a component affects step properties.	Optional	<a href="#">3.6 Setting step properties</a>
4	Set service properties when changing the version of a component affects service properties.	Optional	<a href="#">4.2 Editing and adding service properties</a>
5	Validate the edited service template <sup>#</sup> .	Optional	<a href="#">8. Validating Service Templates</a>
6	Release a completed service template and prepare to create the service.	Mandatory	<a href="#">5.4.2 Procedure for releasing a service template</a>
7	To move the service template from the development environment to another environment, export the service template, and then import it to the destination environment.	Optional	<a href="#">5.5.1 Procedure for exporting service templates</a> , <a href="#">5.6.1 Procedure for importing service templates</a>
8	Create, edit, and execute the service.	Mandatory	<i>JP1/Automatic Operation Administration Guide</i> <ul style="list-style-type: none"><li>• Creating services</li><li>• Editing services</li><li>• Executing services</li></ul>

#

If the validation process finds an issue with the service template, repeat tasks 2 to 5 as needed.

### 1.5.5 General procedure for adding or deleting processing to or from a service template

Users can add processing in the form of steps to a flow in an existing service template. Deleting a step will delete the corresponding processing.

#### You might use this procedure when:

- You want to insert an email-sending process at the end of the processing automated by the service template.
- A file transfer step is no longer required in processing that acquires log data for JP1/IM and JP1/Base.

#### Required knowledge

- [2.1 Overview of development service templates and release service templates](#)
- [3.2 Relationship between flow and steps](#)

- [3.3 Creating flow hierarchies](#)
- [5.4.1 Overview of service template release](#)

## General procedure

Table 1-7: General procedure for adding or deleting processing to or from a service template

Task		Mandatory/ optional	Refer to
1	Copy a release service template that you want to edit.	Optional	<a href="#">5.2.1 Procedure for copying service templates</a>
2	Add or delete steps.	Mandatory	<a href="#">3.4.1 Procedure for adding steps</a> , <a href="#">3.5.3 Operations that can be performed on steps and relational lines</a>
3	Check and (if necessary) change the order of executing steps when adding or deleting a step affects the processing flow.	Optional	<a href="#">3.5.1 Procedure for defining the execution order of steps</a>
4	Set step properties or service properties when adding or deleting a step affects step properties or service properties respectively.	Optional	<a href="#">3.6 Setting step properties</a> , <a href="#">4.2 Editing and adding service properties</a>
5	Validate the edited service template <sup>#</sup> .	Optional	<a href="#">8. Validating Service Templates</a>
6	Release the completed service template and prepare to create the service.	Mandatory	<a href="#">5.4.2 Procedure for releasing a service template</a>
7	To move the service template from the development environment to another environment, export the service template, and then import it to the destination environment.	Optional	<a href="#">5.5.1 Procedure for exporting service templates</a> , <a href="#">5.6.1 Procedure for importing service templates</a>
8	Create, edit, and execute the service.	Mandatory	<i>JP1/Automatic Operation Administration Guide</i> <ul style="list-style-type: none"> <li>• Creating services</li> <li>• Editing services</li> <li>• Executing services</li> </ul>

#

If the validation process finds an issue with the service template, repeat tasks 2 to 5 as needed.

## 1.5.6 General procedure for dynamically or statically setting property values when executing services

By mapping step properties to other step properties or service properties, you can execute processing by assigning the property value. You can also assign a fixed value to an input property.

This process is not limited to existing input properties or output properties. Users can create new properties and map them.

## You might use this procedure when:

- In processing that increases available memory, you want to be able to specify how much memory to allocate when executing a service (dynamically setting input properties), instead of using a fixed value.
- In processing that increases available memory, you want the memory capacity to be fixed at 5 GB each time the service runs (statically setting input properties).

## Required knowledge

- [2.1 Overview of development service templates and release service templates](#)
- [3.6.1 Overview of step properties](#)
- [3.6.7 Example of defining step properties](#)
- [5.4.1 Overview of service template release](#)

## General procedure

Table 1-8: General procedure for dynamically or statically setting property values when executing services

Task		Mandatory/ optional	Refer to
1	Copy a release service template that you want to edit.	Optional	<a href="#">5.2.1 Procedure for copying service templates</a>
2	Set step properties.	Mandatory	<a href="#">3.6 Setting step properties</a>
3	Set service properties when setting a step property affects service properties.	Optional	<a href="#">4.2 Editing and adding service properties</a>
4	Validate an edited service template <sup>#</sup> .	Optional	<a href="#">8. Validating Service Templates</a>
5	Release the completed service template and prepare to create the service.	Mandatory	<a href="#">5.4.2 Procedure for releasing a service template</a>
6	To move the service template from the development environment to another environment, export the service template, and then import it to the destination environment.	Optional	<a href="#">5.5.1 Procedure for exporting service templates</a> , <a href="#">5.6.1 Procedure for importing service templates</a>
7	Create, edit, and execute the service.	Mandatory	<i>JP1/Automatic Operation Administration Guide</i> <ul style="list-style-type: none"><li>• Creating services</li><li>• Editing services</li><li>• Executing services</li></ul>

#

If the validation process finds an issue with the service template, repeat tasks 2 to 4 as needed.

## 1.5.7 General procedure for setting service properties

You can set properties that are used by users who execute services to specify parameters necessary for executing the services and to acquire the execution results of the services.

### You might use this procedure when:

- In processing that increases available memory, you want to be able to specify how much memory to allocate when executing a service, instead of using a fixed value.
- You want to check the processing result for a plug-in in the **Task Details** window.

### Required knowledge

- [2.1 Overview of development service templates and release service templates](#)
- [3.6.1 Overview of step properties](#)
- [3.6.7 Example of defining step properties](#)
- [4.2.1 Overview of service property](#)
- [5.4.1 Overview of service template release](#)

### General procedure

Table 1-9: General procedure for setting service properties

Task		Mandatory/ optional	Refer to
1	Copy a release service template that you want to edit.	Optional	<a href="#">5.2.1 Procedure for copying service templates</a>
2	Set step properties when you want to elevate step properties to service properties.	Optional	<a href="#">3.6 Setting step properties</a>
3	Set service properties.	Mandatory	<a href="#">4.2 Editing and adding service properties</a>
4	Validate an edited service template <sup>#</sup> .	Optional	<a href="#">8. Validating Service Templates</a>
5	Release the completed service template and prepare to create the service.	Mandatory	<a href="#">5.4.2 Procedure for releasing a service template</a>
6	To move the service template from the development environment to another environment, export the service template, and then import it to the destination environment.	Optional	<a href="#">5.5.1 Procedure for exporting service templates</a> , <a href="#">5.6.1 Procedure for importing service templates</a>
7	Create, edit, and execute the service.	Mandatory	<i>JPI/Automatic Operation Administration Guide</i> <ul style="list-style-type: none"><li>• Creating services</li><li>• Editing services</li><li>• Executing services</li></ul>

#

If the validation process finds an issue with the service template, repeat tasks 2 to 4 as needed.

## 1.6 Using existing service templates provided by JP1/AO

This section describes the general procedure for using existing service templates provided by JP1/AO as they are, and gives references to manuals.

### 1.6.1 General procedure for using an existing service template provided by JP1/AO

When appropriate, you can use the service templates provided in the standard JP1/AO package and the JP1/AO Content Set.

#### You might use this procedure in situations like the following:

A template provided by JP1/AO exactly defines the task you want to automate:

- You want to use the Add monitoring setting service template to add multiple monitored servers to HP NNMi or JP1/ PFM.
- You want to use the Add operational user service template to add OS users, JP1 users, and the associated mapping information.

#### Required knowledge

- [2.1 Overview of development service templates and release service templates](#)

#### General procedure

Table 1-10: General procedure for using an existing service template provided by JP1/AO

Task		Mandatory/ optional	Refer to
1	Evaluate the service template you want to use.	Mandatory	<ul style="list-style-type: none"><li>• Evaluating the service template to be used and the targets of operation in the <i>JP1/Automatic Operation Overview and System Design Guide</i></li><li>• <i>JP1/Automatic Operation Service Template Reference</i><ul style="list-style-type: none"><li>• List of JP1/AO Standard-package Service Templates</li><li>• List of JP1/AO Content Pack service templates</li></ul></li></ul>
2	Add service templates to JP1/AO.	Mandatory	Importing service templates in the <i>JP1/Automatic Operation Administration Guide</i>
3	Create, edit, and execute services.	Mandatory	<i>JP1/Automatic Operation Administration Guide</i> <ul style="list-style-type: none"><li>• Creating services</li><li>• Editing services</li><li>• Executing services</li></ul>

## 1.7 List of service template development features

The following table lists the features of JP1/AO that are used in the development of service templates.

Table 1-11: List of service template development features

Feature		Description	Refer to
Creating and editing service templates	Creating and editing definition information	Users can create new original service templates. Users can also edit service templates and change information (such as, the service template names, descriptions, service properties, and custom files).	<a href="#">2.2 Creating and changing the service template definition information</a>
	Adding steps	Users can add steps to a flow to place necessary processing in a task. Users can also edit the definition information of steps.	<a href="#">3.4 Adding and editing steps</a>
	Defining the execution order of steps	Users can add steps and connect them by relational lines to define execution order of the processing.	<a href="#">3.5 Defining the execution order of steps</a>
	Setting step properties	Users can dynamically or statically set the property values used when services are executed.	<a href="#">3.6 Setting step properties</a>
	Setting service properties	Users can define the input items when setting or executing services, or the items displayed when checking the execution result.	<a href="#">4.2 Editing and adding service properties</a>
	Managing the versions of components used as steps	Users can check and change the version of the components used as steps.	<a href="#">3.7 Managing the versions of components used as steps</a>
	Setting display information of service templates	Users can set the display information of service templates (such as service template names and descriptions) in resource files. Note that resource files can be set for each locale for the Web browser.	<a href="#">2.3 Setting display information for service templates in resource files</a>
Managing service templates	Viewing service templates	Users can view the definition information and detailed processing of service templates.	<a href="#">5.1 Viewing service templates</a>
	Copying service templates	Users can copy service templates. Users can also edit the copied service templates and create new service templates.	<a href="#">5.2 Copying service templates</a>
	Deleting service templates	Users can delete service templates.	<a href="#">5.3 Deleting development service templates</a>
	Releasing service templates	Releasing validated service templates creates the packages of the service templates, and they are imported to the JP1/AO server. Service templates after being released, cannot be edited. To move service templates from the development environment to another environment, users must manually import the service templates.	<a href="#">5.4 Releasing service templates</a>
	Exporting service templates	Users can store service template files in any folder.	<a href="#">5.5 Exporting service templates</a>
	Importing service templates	Users can import service templates to the destination environment when moving the released service templates from the development environment to another environment.	<a href="#">5.6 Importing service templates</a>
Creating and editing plug-ins	Creating plug-ins	Users can create new original plug-ins.	<a href="#">6.2 Creating and editing plug-in definition information</a> , <a href="#">6.3 Setting plug-in properties</a> , <a href="#">6.4 Editing platforms</a>
	Editing plug-ins	Users can edit plug-ins, by setting the plug-in names, input properties, output properties, and remote commands. Note that users cannot edit plug-ins provided by JP1/AO.	

Feature		Description	Refer to
Creating and editing plug-ins	Setting display information of plug-ins	Users can set the display information of plug-ins (such as, plug-in names and descriptions) in resource files. Note that users can set resource files for each locale of the Web browser.	<a href="#">6.5 Using resource files to set plug-in display information</a>
Managing plug-ins	Copying plug-ins	Users can copy development plug-ins and release plug-ins, and create new plug-ins by editing the copied plug-ins.	<a href="#">7.1 Copying plug-ins</a>
	Deleting plug-ins	Users can delete development plug-ins and release plug-ins.	<a href="#">7.2 Deleting plug-ins</a>
Validating service templates	Building service templates	Building service templates creates packages for the service templates being developed, and then they are imported to the JP1/AO server. This processing is performed to validate service templates.	<a href="#">8.2 Building service templates</a>
	Debugging service templates	Users can check the behavior of service templates that were built, and find problems. If debugging finds a problem in flows or plug-ins, users can edit the service templates or plug-ins.	<a href="#">8.3 Debugging service templates</a>
	Conducting operation tests for service templates	Users can create and execute services from the service templates that were built, and find operational problems. If operational tests find a problem, users can edit the service templates or plug-ins.	<a href="#">8.5 Testing the operation of service templates</a>



# 2

## Setting Service Template Definition Information

This chapter describes how to set definition information necessary for creating and editing service templates.

## 2.1 Overview of development service templates and release service templates

---

Service templates consist of development service templates and release service templates.

### Development service template

A development service template is a service template a user is developing. Service templates created by copying a release service template are also categorized as development service templates.

When you build a development service template, Debug is set as the configuration type and execution of the service can be tested. Services created from a development service template are used in a development environment. Any service template that is not yet built is also categorized as a development service template.

Development service templates are displayed in the following windows:

- **Developing** tab of the **Service Builder Home** window
- **Select Service Template** dialog box (which users in a Develop or higher role can display by clicking the **Create** button in the **Services** window)

### Release service template

A release service template is a service template that has been imported into the JP1/AO server by releasing a development service template. Service templates provided by JP1/AO are also categorized as release service templates. Release service templates are used for real-world applications in the active environment. Release is set as the configuration type of release service templates.

Service templates that have been imported to the JP1/AO server and have the configuration type Release are handled as release service templates.

Release service templates are displayed in the following windows:

- **Service Template** window
- **Released** tab of the **Service Builder Home** window
- **Select Service Template** dialog box (which can be displayed by clicking the **Create** button in the **Services** window)

Note that you cannot edit a service template after its release. To edit such a template, copy the release template and then edit the copy as a development service template.

### Important

Once edited, the service templates and plug-ins provided by JP1/AO are outside the scope of JP1/AO product support. However, product support is still offered for the plug-ins provided by JP1/AO (in the standard package or the JP1/AO Content Set) that are called from such templates.

---

### Related topics

- [5.4.2 Procedure for releasing a service template](#)
-

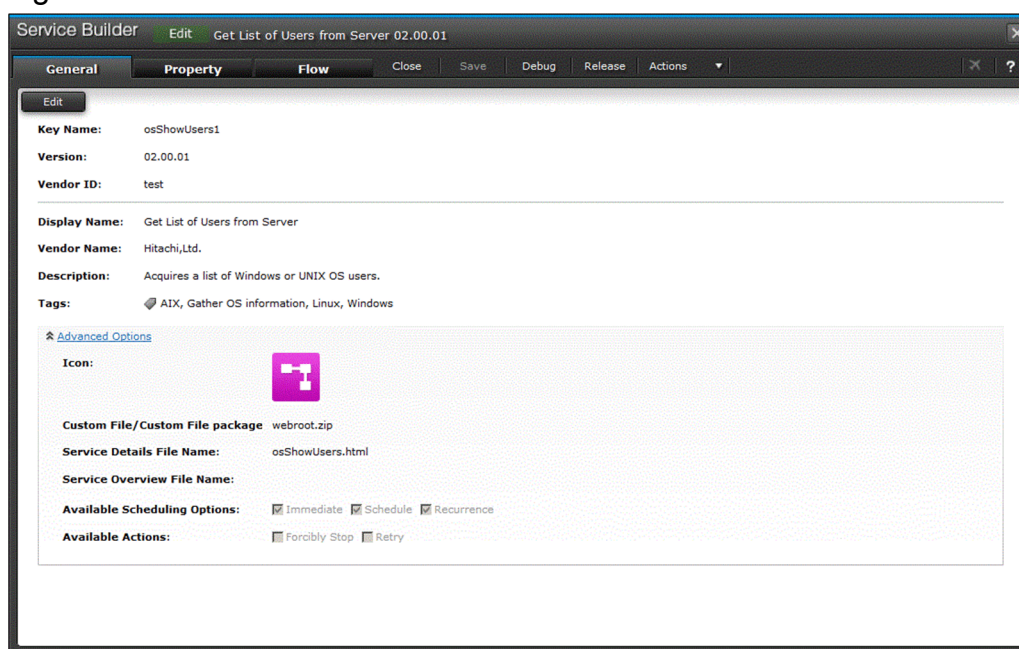
## 2.2 Creating and changing the service template definition information

After you create a blank service template or copy an existing release service template, set the definition information for the service template.

### 2.2.1 Service Builder Edit window General tab

On the **General** tab of the **Service Builder Edit** window, you can edit the definition information for a created or copied service template.

Figure 2-1: **Service Builder Edit** window **General** tab



This tab displays the definition information for a service template. You can edit the information by clicking the **Edit** button.

### 2.2.2 Procedure for creating blank service templates

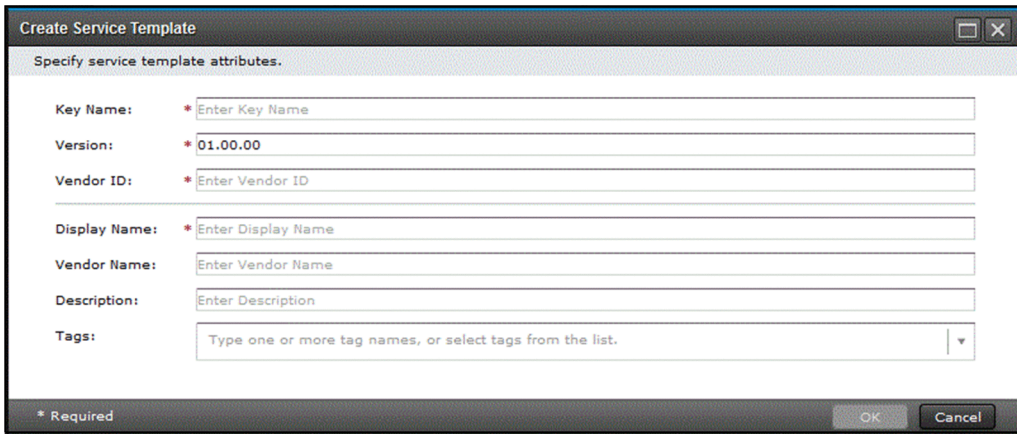
The first thing you need to do when creating a new service template is set the service template ID, service template version, vendor ID, and other definition information.

#### To create a blank service template:

1. On the **Developing** tab of the **Service Builder Home** window, click the **Create** button.
2. In the **Create Service Template** dialog box, set the definition information for the service template, and then click the **OK** button.

Note that you cannot change the values in the **Service Template Key Name**, **Service Template Version**, and **Vendor ID** fields after the service template has been created. Make sure you confirm the definition information before saving it.

Figure 2-2: **Create Service Template** dialog box

The image shows a 'Create Service Template' dialog box with a title bar containing a minimize button, a maximize button, and a close button. The main area is titled 'Specify service template attributes.' and contains several input fields: 'Key Name:' with a red asterisk and placeholder 'Enter Key Name'; 'Version:' with a red asterisk and placeholder '01.00.00'; 'Vendor ID:' with a red asterisk and placeholder 'Enter Vendor ID'; 'Display Name:' with a red asterisk and placeholder 'Enter Display Name'; 'Vendor Name:' with placeholder 'Enter Vendor Name'; 'Description:' with placeholder 'Enter Description'; and 'Tags:' with a text area containing the placeholder 'Type one or more tag names, or select tags from the list.' and a dropdown arrow. At the bottom left, there is a legend '\* Required'. At the bottom right, there are 'OK' and 'Cancel' buttons.

### Tip

You can also create a blank service template, by clicking the **Create Service Template** button in the **Service Template** window.

## Operation result

A blank service template is created, and the **Flow** tab of the **Service Builder Edit** window is displayed.

## Related topics

- [2.2.4 Items to set in service template definition information](#)
- [1.2.3 Procedure for starting editing of service templates](#)
- [2.2.3 Procedure for changing the service template definition information](#)
- [3. Creating and Editing Flows for Service Templates](#)

## 2.2.3 Procedure for changing the service template definition information

After creating or copying a service template, in the **General** tab of the **Service Builder Edit** window, you can change the previously specified definition information of the service template.

### To edit service template definition information

1. On the **Developing** tab of the **Service Builder Home** window, select the development service template you want to edit, and then click the **Edit** button.
2. On the **General** tab of the **Service Builder Edit** window, click the **Edit** button.
3. In the **Edit Service Template Attributes** dialog box, set the definition information of the service template.

Figure 2-3: **Edit Service Template Attributes** dialog box

Edit Service Template Attributes

Specify service template attributes.

Key Name:

getInfoVMhyperV

Version:

02.00.10

Vendor ID:

hitachi

Display Name:

\* Obtain the virtual server information list

Vendor Name:

Hitachi, Ltd.

Description:

Obtains the virtual server information list in the Hyper-V environment.


Tags:

Gather VM information

Hyper-V 2008

Advanced Options

Icon:



Restore Default Icon

Change

Custom File/Custom File package:

webroot.zip

Browse

Delete

Service Details File Name:

getInfoVMhyperV.html

Service Overview File Name:

Enter Service Overview File Name

Available Scheduling Options:

Immediate

Schedule

Recurrence

Available Actions:

Forcibly Stop

Retry

\* Required

OK

Cancel

4. Click the **OK** button.

## Operation result

Definition information of the service template is set.

!

Important

If another user has performed an intervening action while you were editing the service template, the save processing might fail.

## Related topics

- 2.2.4 Items to set in service template definition information
- 1.2.3 Procedure for starting editing of service templates
- 1.2.4 Notes when an interrupt operation is performed in the **Service Builder** window

## 2.2.4 Items to set in service template definition information

In the **Create Service Template**, **Copy Service Template**, and **Edit Service Template Attributes** dialog boxes, you can set the following items.

Table 2-1: Items to set in service template definition information

Item	Description
Service Template Key Name <sup>#1</sup>	Specify the ID used to identify the service template.
Service Template Version <sup>#1</sup>	Specify the version number of the service template in aa.bb.cc format.
Vendor ID <sup>#1</sup>	Specify the ID used to identify the vendor who created the service template.

Item	Description
<b>Vendor ID</b> <sup>#1</sup>	Create a unique vendor ID by specifying the domain name in reverse order from the top level as a period-separated value. For example, specify vendor IDs in the format com.xxxx or jp.co.yyyy. If you choose not to use domain names as vendor IDs, make sure that the vendor ID you specify is not being used for another vendor. Note that you cannot specify a vendor ID that begins with com.hitachi.software.dna.
<b>Service Template Name</b>	Specify the name of the service template.
<b>Vendor Name</b> <sup>#2</sup>	Specify the name of the vendor that created the service template.
<b>Description</b>	Specify the description of the service template.
<b>Tags</b>	Specify one or more tags to be defined for the service template.

#1

You cannot change **Service Template Key Name**, **Service Template Version**, and **Vendor ID** after the service template is created or copied. The uniqueness of a service template is guaranteed by the combination of these three items.

#2

If you omit specifying this item, the value specified for **Vendor ID** is set for **Vendor Name**. In this case, note that **Vendor Name** of the development service template displayed in the window remains blank.

In the **Edit Service Template Attributes** dialog box, you can also set the following items as advanced options.

Table 2-2: Items that can be set as advanced options

Item	Description
Icon	The icon set for the service template is displayed. Clicking the <b>Back to the Default</b> button changes the icon set for the service template back to the default. Clicking the <b>Change</b> button displays the dialog box where you can select the icon file to be uploaded and change the icon. For the icon, set the file in png format (48 x 48 pixels).
Custom Files	The custom files set for the service template are displayed. Clicking a file name enables you to download the file. Clicking the <b>Select</b> button displays the dialog box for selecting a folder, where you can set custom files. Clicking the <b>Delete</b> button cancels the specified custom files.
Service Details File Name	If a file in zip format has been specified for a custom file, in the <b>Service Detail(s)</b> window, specify the relative path to the settings file.
Service Overview File Name	If a file in zip format has been specified for a custom file, specify the relative path to the image file for service overview.
Available Scheduling Options	Specify the options for the schedule types that can be specified when the service is executed. Schedule types include Immediate, Scheduled, and Recurring. You can further narrow down the options for the schedule types when creating or editing a service.
Available Actions	Specify whether to permit Forcibly Stop and Retry actions for tasks that use the service template. You can further narrow down the available actions when creating or editing a service.

## Related topics

- [5.2.2 Uniqueness of service templates and plug-ins](#)
- [6.2.4 Image files that can be set for component icons](#)
- [2.2.5 Overview of custom files to be set to service templates](#)



## 2.2.5 Overview of custom files to be set to service templates

A custom file defines the contents and format of a service displayed in the window. By setting custom files in the **Edit Service Template Attributes** window, you can define the contents displayed in the **Service Detail(s)** window and the image file for service overview. By setting custom files for individual locales of the Web browser, you can view the service details in the language corresponding to the locale of the Web browser.

The image file for service overview is displayed in the following windows:

- **Service Definition** window
- **Submit Service** window
- **Task Details** window

The following figures show examples of windows.

Figure 2-4: Example of the **Service Detail(s)** window

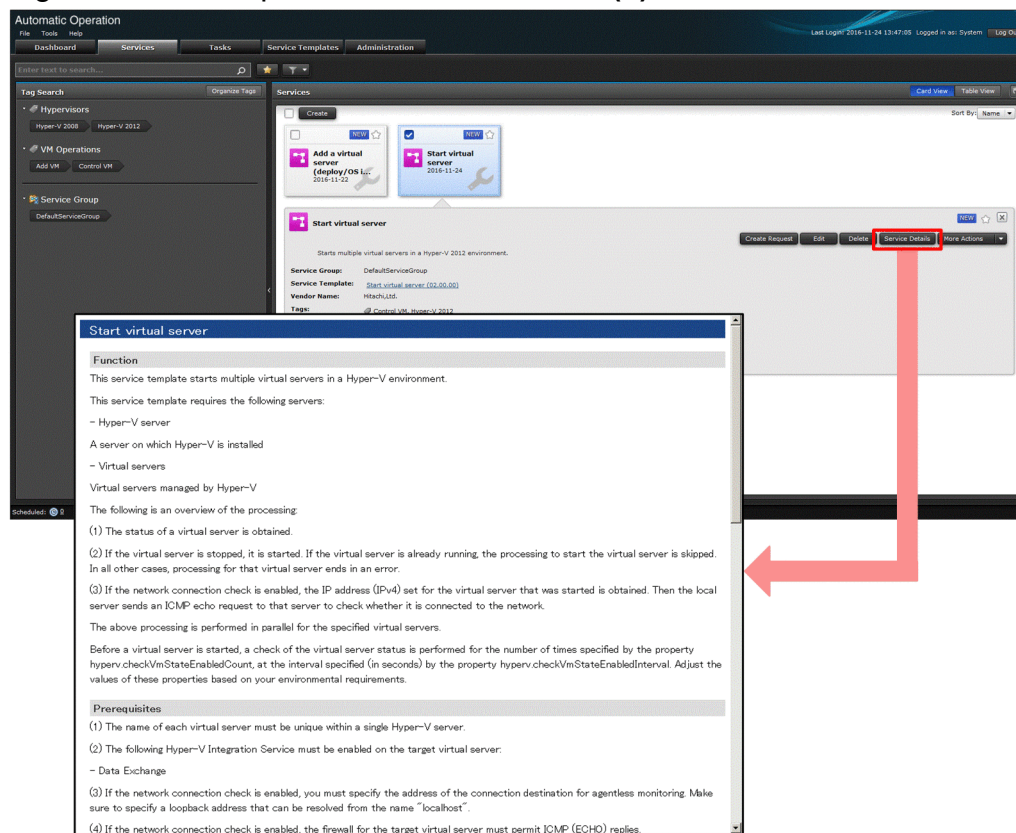
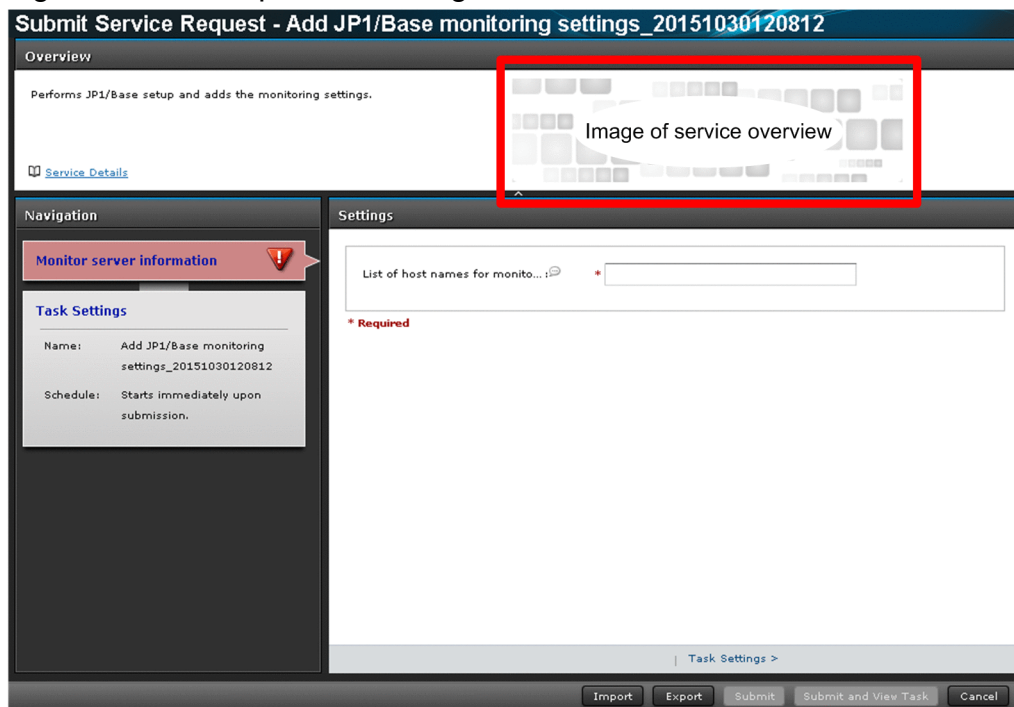


Figure 2-5: Example of the image file for service overview



## Related topics

- [2.2.6 Procedure for setting custom files for service templates](#)
- [2.2.7 Switching custom files for individual locales of the Web browser](#)
- [2.2.8 Format of custom files](#)

## 2.2.6 Procedure for setting custom files for service templates

You can set one or more files as the custom files for a service template so that, for example, you can set figures and links, compress multiple files or folders in zip format, and then register the zip file.

Note that you can change custom files even after the setup.

### To set custom files:


1. Store the multiple files you created under the *any-folder*\webroot folder.<sup>#</sup>
2. Compress the files under the webroot folder in zip format. Do not include the webroot folder itself in the zip folder.<sup>#</sup>
3. In the **Edit Service Template Attributes** dialog box, under **Custom Files**, click the **Select** button.



Figure 2-6: **Edit Service Template Attributes** dialog box

4. Select the file you want to specify for the custom file.
5. In the **Service Details File Name** or **Service Overview File Name** text box, enter the relative path to the corresponding file.<sup>#</sup>
- Set the path that the zip file you specified has been expanded to as the current path, and then specify the relative path. Use a slash (/) as the delimiter of the path.

<sup>#</sup>: This step is required for setting multiple custom files.

 **Tip**

If the extension of a file is .zip, after the file is built or released, the file name is automatically changed to webroot.zip.

## Operation result

Custom files are set for the service template.

### Related topics

- [2.2.5 Overview of custom files to be set to service templates](#)
- [2.2.7 Switching custom files for individual locales of the Web browser](#)

## 2.2.7 Switching custom files for individual locales of the Web browser

When creating custom files in zip format, store the files for each locale under the corresponding webroot\language-code folder. Then the custom files corresponding to the locale of the Web browser are loaded, and the display is switched according to the custom files.

Compress the files under the webroot folder in zip format. However, for the **Service Details File Name** or **Service Overview File Name** text box, set the webroot\language-code folder as the current path, and specify the relative path.

You can specify, as the language code, two-digit lowercase letters (ja, en, or zh) as defined in ISO-639.

If you also store custom files directly under the webroot folder, those files are loaded in the cases below. Thus, for example, you can set it to load English custom files when there is no custom file that corresponds to the locale of the Web browser.

- The language code folder corresponding to the locale of the Web browser does not exist.
- No file exists in the language code folder corresponding to the locale of the Web browser.

### Important

- The file names directly under the webroot folder must be the same as the file names in the language code folder.
- Even when you do not store files directly under the webroot folder, specify the same file names for the files under individual language folders.

---

## Related topics

- [2.2.6 Procedure for setting custom files for service templates](#)
  - [2.2.8 Format of custom files](#)
- 

## 2.2.8 Format of custom files

Create custom files as static content to be executed in a Web browser.

### Specifiable extensions

The following are some of the extensions you can specify for custom files:

Custom files that define the contents displayed in the **Service Detail(s)** window

- .html
- .js
- .css
- .swf
- .jpeg

Custom files that define the image for service overview

.png

We recommend that you use an image file of 150 x 420 pixels (vertical x horizontal).

Note that JP1/AO product support does not extend to dynamic content such as .jsp and .war files that run on an application server.

### Characters specifiable in file names and paths

Use ASCII characters in the file names and paths of custom files. You cannot use the following characters:

- Multi-byte characters

- Control characters ('\u0000' to '\u001F' and '\u007F' to '\u009F')
- Question marks (?), asterisks (\*), double quotation marks ("), right angle brackets (>), left angle brackets (<), vertical bars (|), and colons (:)

## 2.3 Setting display information for service templates in resource files

You can assign resource files to service templates, and define different information to display on screen for different Web browser locales. For example, you can have the service template name displayed in a language that is appropriate for the locale of the Web browser.

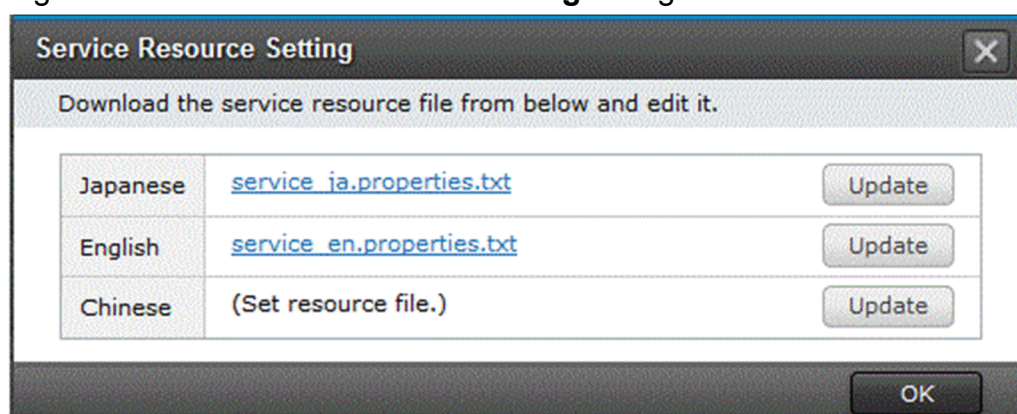
### 2.3.1 Procedure for setting service resource files

By assigning a service resource file, you can define the information displayed in a service template. You can then set display information for service templates, steps, and service properties. To edit a service resource file, you download the file and overwrite the contents as needed.

Note that you cannot change the information displayed for a release service template unless you copy the release service template and edit it as a development service template.

1. In the **Service Builder Home** window, click the **Developing** tab, select the service templates that you want to edit, and then click the **Edit** button.
2. In the **Service Builder Edit** window, from the **Actions** pull-down menu, select **Set Resources**.
3. In the **Service Resources Setting** dialog box, click the link for the service resource file, and download the service resource file.

Figure 2-7: **Service Resources Setting** dialog box



4. Edit the definitions in the service resource file you downloaded.  
Do not change the file name from `service_<language-code>.properties.txt`. If you change the file name, an error will occur when you attempt to upload the file.  
For the *language-code*, you can specify two-digit lowercase letters (ja, en, or zh) as defined in ISO-639.
5. Click the **Refresh** button, select the service resource file you edited, and then upload the file.
6. In the confirmation dialog box, click the **OK** button.



#### Important

When you upload a service resource file, the existing file is overwritten with the contents of the new file. Take care not to upload the wrong file.

### Operation result

Display information of the service template is set, according to the contents of the resource file.

---

## Related topics

- [5.2.1 Procedure for copying service templates](#)
  - [2.3.2 Format of service resource file](#)
  - [2.3.3 Definitions in service resource files](#)
- 

## 2.3.2 Format of service resource file

A service resource file defines the items displayed in the operation windows of JP1/AO. The format of the file is described below. Note that the definitions in the service resource file depend on the type of display items you are defining. For details about how to define each type of display item, see [2.3.3 Definitions in service resource files](#).

- The file name of the service resource file is `service_ language-code.properties.txt`.  
As *language-code*, you can specify two-digit lowercase letters (ja, en, or zh) as defined in ISO-639.
- Define the file contents in the format *property-key delimiting-character setting-value*. As the delimiting character, you can use an equals sign (=), a colon (:), tab characters (\t), or a single-byte space.
- Enter one property key and setting per line.
- Property keys can contain the following characters:
  - Single-byte alphanumeric characters
  - Single-byte hyphens (-)
  - Single-byte underscores (\_)
  - Single-byte periods (.)
- Characters must be encoded in UTF-8
- If you define the same property key in the file more than once, the value of the last occurrence of the property key applies.
- Lines that begin with a hash mark (#) are handled as comments.
- Property keys are case sensitive.
- To specify a character string that contains a back slash (\), specify two back slashes (\\) instead.
- Lines that consist only of single-byte spaces are ignored.
- On each line of the service resource file, the property key is the character string from the first character that is not a single-byte space to the character immediately preceding the first delimiting character.
- The setting value is the string from the first non-delimiting character after the delimiting character following the property key to the last character in the line.  
For example, the following line in the service resource file represents the property key abc with the setting value = \tc.  
`abc\t=\tc`  
However, if the character immediately following the first delimiting character is = or :, the setting value is the character string from the next character that is not a single-byte space or tab character (\t), to the end of the line.  
For example, the following line in the service resource file represents the property key abc with the setting value = \tc.  
`abc\t=\t=\tc`
- You cannot use surrogate pair characters.

### 2.3.3 Definitions in service resource files

The definitions in the service resource file depend on the type of display items you are setting. You can specify the following definitions in the service resource file:

#### Setting items displayed in the **Edit Service Template Attributes** dialog box

Define entries in the following format to set the service template names, descriptions, and other information displayed in the **Edit Service Template Attributes** dialog box.

*property-key delimiting-character setting-value*

Property keys can be 1 to 128 characters long.

For example, enter a definition in the format `service.displayName=This is a test service..`

#### When specifying display items for a step

Define entries to set the step name and description. The step name and description are displayed in the **Flow** area of the **Tasks** window when you execute the service.

##### Setting a step name

To set a step name in a service template, enter a definition in the following format:

*dnajob.step-ID.displayName delimiting-character setting-value*

For example, enter a definition in the format `dnajob.teststep.comment=This is a test step..`

##### Setting a step description

To set a description of a step, enter a definition in the following format:

*dnajob.step-ID.comment delimiting-character setting-value*

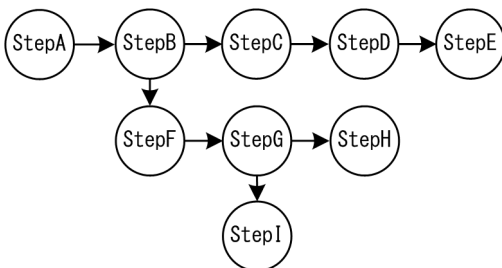
For example, enter a definition in the format `dnajob.teststep.comment=This is a test step..`

#### When specifying display items for a step that is not on the highest hierarchical level of a flow

Specify the step IDs from the step at the highest hierarchical level to the target step, connecting them with slashes (/).

For example, to specify a resource in Step F in the figure below, connect the step ID for Step B at the highest hierarchical level to the step ID for Step F. To specify a resource in Step I, connect the step IDs of Step B, Step G, and Step I.

Figure 2-8: Example of specifying resources in Step F and Step I



The following shows how to specify resources in Step F and Step I in a service resource file.

```
dnajob.StepB/StepF.displayName=Step F
dnajob.StepB/StepF.comment=Step F description
dnajob.StepB/StepG/StepI.displayName=Step I
dnajob.StepB/StepG/StepI.comment=Step I description
```

---

## Related topics

- [2.3.4 Correspondence between information displayed in service templates and properties in service resource files](#)
- 

### 2.3.4 Correspondence between information displayed in service templates and properties in service resource files

The display information for service templates can be set in the window. The display information you set in the window is also defined in the resource file for the service. The following table lists the correspondence between the information displayed for a service template and the properties in the service resource file.

Table 2-3: Correspondence between information displayed for service templates and properties in service resource files

Information displayed for service template	Property in service resource file
Vendor name	service.vendorDisplayName
Service template name	service.displayName
Service template description	service.shortDescription
Property group name	When there is no related step in the property group propertyGroup. <i>property-group-ID</i> .displayName When there is a related step in the property group propertyGroup. <i>step-ID/property-group-ID</i> .displayName <sup>#</sup>
Property group description	When there is no related step in the property group propertyGroup. <i>property-group-ID</i> .description When there is a related step in the property group propertyGroup. <i>step-ID/property-group-ID</i> .description <sup>#</sup>
Service property name	When there is no related step in the property group property. <i>property-key</i> .displayName When there is a related step in the property group property. <i>step-ID/property-key</i> .displayName <sup>#</sup>
Service property description	When there is no related step in the property group property. <i>property-key</i> .description When there is a related step in the property group property. <i>step-ID/property-key</i> .description <sup>#</sup>
Step property name	property. <i>step-ID/property-key</i> .displayName <sup>#</sup>
Step property description	property. <i>step-ID/property-key</i> .description <sup>#</sup>
Step name	dnajob. <i>step-ID</i> .displayName <sup>#</sup>
Step description	dnajob. <i>step-ID</i> .comment <sup>#</sup>

#

To set a resource for a step that is not at the highest hierarchical level of a flow, specify the step IDs from the step at the highest hierarchical level to the target step, connecting them with forward slashes (/).



Information of the step property is displayed under the dotted line of the service resource file. After uploading a service resource file, if you elevate a step property to a service property in the **Flow** tab of the **Service Builder Edit** window, the information already set for the step property is applied to the information of the service property.

### 2.3.5 Service resource files automatically generated when a service template is created

When a service template is created, two service resource files are automatically generated: one is for the same language as the Web browser locale, and the other is for English. However, if the locale of the Web browser is English, only the one for English is generated.

The following table lists the values set in these automatically generated service resource files.

Table 2-4: Default values for display information set in service resource files (at service template creation)

Defined display information	Value set by default	
	Same language as Web browser locale	Automatically generated English language resource file#
Vendor name	The value specified in the <b>Service Builder</b> window	Vendor ID
Service template name		Service template ID
Service template description		Blank

#  
For the contents of the service resource file generated when the Web browser locale is English, see the *Same language as the Web browser locale* column.

Examples of service resource files automatically generated when creating a service template are shown below.

#### Values specified in the window for creating a service template

```
Vendor ID: test.vendor
Service template ID: test.service
Service template version: 10.00.00
Service template name: test.template
Vendor name: test.vendor
Description: This service template is for testing purposes.
```

#### Generated service resource file

```
service.vendorDisplayName=test.vendor
service.displayName=test.service
service.shortDescription=This service template is for testing purposes.
```



## 2.3.6 Service resource files updated when a service template is edited

When you edit and save a service template, JP1/AO updates the service resource file for the same language as the Web browser locale.

However, in the following cases, corresponding changes are made to the other service resource files for the non-Web browser locale language to ensure consistency:

- A definition of display information is added or deleted.
- A property group ID, step ID, or property key is updated.

The table below shows the values added to the service resource file for locales other than the Web browser locale when you add a definition of a display item. When you update the definition of a display item and update a property group ID, property key, or step ID, the values in the file are automatically overwritten with the values in the table. If you want to reference the existing value, create a backup of the service resource file for locales other than the Web browser locale.



### Important

When the step ID of a layering step or repeated step is updated, the names and descriptions of subordinate steps are automatically overwritten with the values in the table below. Note that the overwritten values are those defined in the service resource file for locales other than the Web browser locale.

Table 2-5: Display item values set in service resource files

Defined display information	Assigned value
Property group name <sup>#1</sup>	Property group ID
Property group description <sup>#1</sup>	Blank
Service property name <sup>#2</sup>	Property key
Service property description <sup>#2</sup>	Blank
Step name <sup>#3</sup>	Step ID
Step description <sup>#3</sup>	Blank

#1

The value of this display item is overwritten when the property group ID is updated.

#2

The value of this display item is overwritten when the property key is updated.

#3

The value of this display item is overwritten when the step ID is updated.

When you delete the definition of a display item, the definition is also deleted from the service resource file for languages other than that of the Web browser locale.

To set display information for a language other than that of the Web browser locale, you need to manually create or edit a service resource file and then upload the file. When manually creating a service resource file, we recommend that you download and use a service resource file for a locale in which display information is already defined.

### **2.3.7 Displaying a service template in a Web browser that is set to a locale for which no service resource file is available**

When you display a service template in a Web browser that is set to a locale for which no service resource file is available, the system uses the service resource file for English language locales to display the template.

# 3

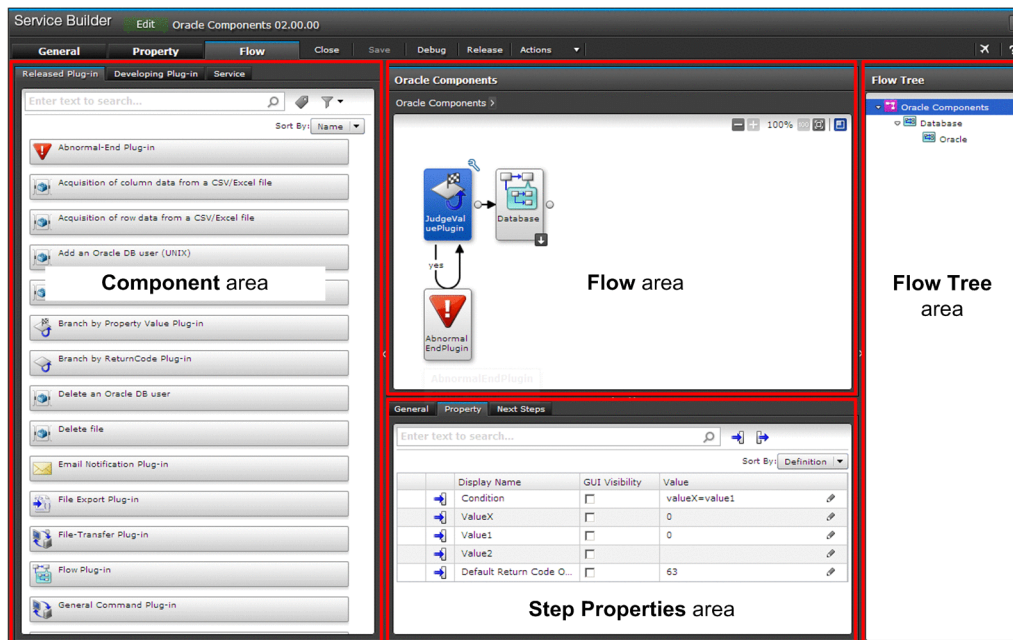
## Creating and Editing Flows for Service Templates

This chapter describes how to create and edit flows for service templates. By creating and editing flows, you can change the order in which the steps are performed, or add steps to an existing procedure.

## 3.1 Service Builder Edit window Flow tab

The **Flow** tab of the **Service Builder Edit** window is used to create and edit flows for service templates. If you perform a create or edit operation in the **Service Builder Home** window, the **Flow** tab of the **Service Builder Edit** window is displayed.

Figure 3-1: **Service Builder Edit** window **Flow** tab



The following describes the displayed items:

### Component area

Displays the components that can be deployed as steps in a flow. If you click the **Release** tab, **Develops** tab, or **Services** tab, the type of the components to be displayed is switched. When you select a component, detailed information about the component is displayed, and buttons appear that you can use to perform operations on the component. You can search for a component by using the search window on the top or by selecting a tag.

### Release tab

Clicking this tab displays release plug-ins.

### Develops tab

Clicking this button displays development plug-ins.

### Services tab

Clicking this button displays service components. If you select a service component and then click the **Service Details** button, you can view a detailed description of the release service template on which the service component is based.

### Flow area

This area defines the order in which the steps are executed.

### Flow Tree area

In this area, you can select the hierarchy of the flow to be displayed in the **Flow** area. The flow names defined in the service template are displayed hierarchically. If you click a flow name, the subordinate flows are displayed in the **Flow** area.

## Step Properties area

In this area, you can set the values for step properties and their visibility. The definition information, a list of properties, and a list of subsequent steps are displayed for the selected step.

### General tab

Edit the definition information of a step.

You can edit the definition information for a step in the **Edit Step** dialog box that appears when you click the **Edit** button. For details, see [2.2.1 Service Builder Edit window General tab](#).

### Property tab

A list of properties of the plug-ins used as steps is displayed. For details, see [4.1 Property tab of the Service Builder Edit window](#).

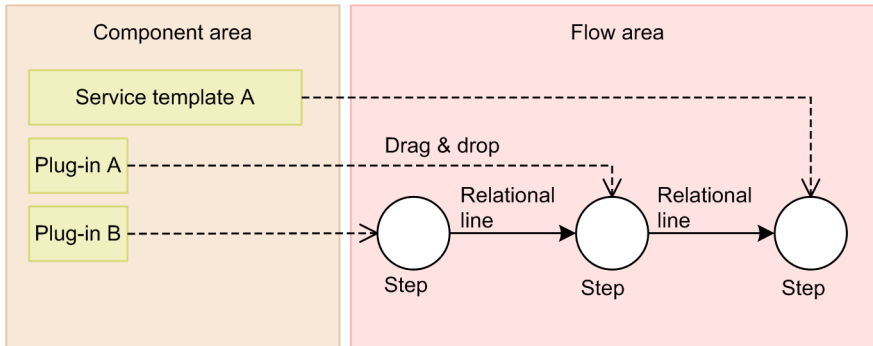
### Next Steps tab

This tab displays a list of subsequent steps for the selected step and a conditional expression that uses arrows to connect the selected step and the subsequent steps. For details, see [3.4.5 Conditional expressions that use arrows to indicate a connection with subsequent steps](#).

## 3.2 Relationship between flow and steps

Plug-ins and service templates that can be deployed in a flow are called *components*. Components are displayed in the **Component** area on the **Flow** tab of the **Service Builder Edit** window. The user adds each unit of processing to a flow by dragging components to the **Flow** area. Each component dropped into the **Flow** area is called a *step*. A flow is created by placing the steps required to execute a task and connecting them with relational lines. The following figure shows the relationship between a flow and steps.

Figure 3-2: Relationship between flow and steps



You can also use flow plug-ins and repeated-execution plug-ins to define a flow within another flow.

### Related topics

- [3.3 Creating flow hierarchies](#)

### 3.3 Creating flow hierarchies

A flow hierarchy is created when you define a flow within another flow. You can define a maximum of 25 hierarchical levels, with the top-level flow being level 1. If you deploy service components as steps, the hierarchical levels in the service components must also be counted for the total number of hierarchical levels. To check the number of hierarchical levels contained in a service component, see the release service template that the service component is based on.

You can create a flow hierarchy by deploying flow plug-ins, and repeat a unit of processing that consists of several steps by deploying repeated execution plug-ins.

Figure 3-3: Creating flow hierarchies

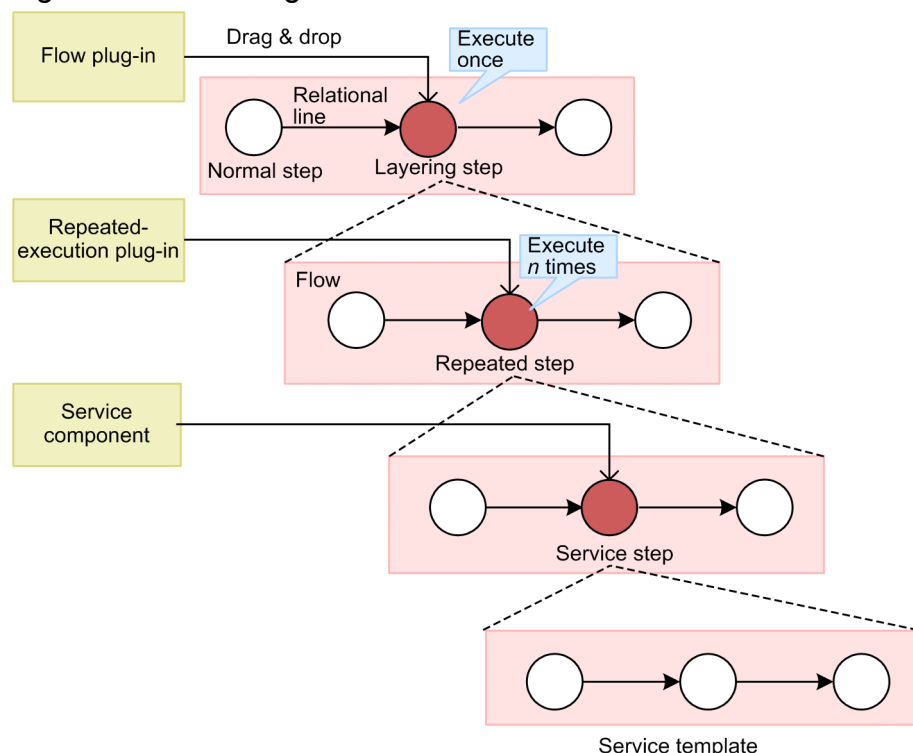


Table 3-1: Component roles and their relationship to steps

Dragged & dropped component	Type of step	Role
Flow plug-in	Hierarchical step	The system creates a flow hierarchy.
Repeated execution plug-in	Repeated step	The system repeats execution of the specified flow. To create a hierarchy under a repeated step, use a flow plug-in or a repeated step. Note that the maximum number of nested levels that can be specified for a repeated execution plug-in is three.
Service component	Service step	The system places a release service template as a component in a flow, and executes the processing.
Other components	Normal step	The system executes the component.



#### Tip

In the **Flow Tree** area on the right side of the **Flow** tab of the **Service Builder Edit** window, you can check the list of defined flows in tree format. In the **Flow Tree** area, the top-level flow is displayed with its service

template name. Each subordinate hierarchy level is displayed with the name of the step that executes the flow plug-in, repeated execution plug-in, or service component.

---

## Related topics


- Task log details in the JP1/Automatic Operation Administration Guide
  - [5.1 Viewing service templates](#)
-



## 3.4 Adding and editing steps

---

You can add steps to a flow, or edit existing steps.

You can view the workflow for adding a step by clicking the Tour icon (  ). This workflow is shown automatically the first time you open the **Flow** tab of the **Service Builder Edit** window.

### 3.4.1 Procedure for adding steps

You can add processing to a flow by adding steps. A step is a component that the user has placed in a flow.

The limits below apply to the number of steps you can add. Do not exceed the maximum number of steps including the number of steps in service components. To check the number of steps in a service component, see the release service template that the service component is based on.

- Maximum number of steps in one service template: 320
- Maximum number of steps at a given hierarchical level: 80



#### Tip

A mini map is displayed when there are too many steps to fit on the screen. For example, a mini map might appear when you are adding a large number of steps, or a large number of steps were added to a given hierarchical level. You can use this mini map to see which part of the overall flow is displayed.

#### To add a step:

1. In the **Component** area on the **Flow** tab of the **Service Builder Edit** window, select the component you want to add as a step.
2. Drag the component you selected to the **Flow** area.
3. In the **Create Step** dialog box, enter the definition information for the step, and then click the **OK** button.

Figure 3-4: **Create Step** dialog box

Create Step

Specify step attributes.

Step ID: \*osReadCSVExcelFileColumn

Step Name: \*Acquisition of column data from a CSV/Excel file

Description:

Component

Version: 02.00.01 (Latest)

**Acquisition of column data from a CSV/Excel file** View

Acquires the data in the specified column from a CSV or Excel file.

**Vendor Name:** Hitachi, Ltd.

**Version:** 02.00.01

**Key Name:** osReadCSVExcelFileColumn

**Vendor ID:** com.hitachi.software.dns.cts.jp1

**Tags:** Gather OS information, Linux, Windows

**Registered:** 2016-11-28 14:56:33

**Last Updated:** 2016-11-28 14:56:33

Next Step Conditions

Condition: Determine by Threshold

Error Threshold: \*0

Use Warnings: ☐

Warning Threshold:

\* Required

OK Cancel

## Operation result

The step is added to the **Flow** area.



### Tip

You can add the following plug-ins to the **Flow** area by selecting **Add step** from the right-click menu in the **Flow** area.

- Branch by property value plug-in
- Branch by returncode plug-in
- Abnormal-end plug-in
- Repeated execution plug-in
- Flow plug-in

## Related topics

- 1.2.3 Procedure for starting editing of service templates
- 3.4.3 Settings in step definition information
- 5.1 Viewing service templates

## 3.4.2 Procedure for editing steps

Editing a step is an operation of changing the definition information that was set in the **Create Step** dialog box when the step was created. You can change the definition information for a step in the **Edit Step** dialog box.

### To edit a step:

1. In the **Flow** area on the **Flow** tab of the **Service Builder Edit** window, right-click the step whose definition information you want to change, and select **Edit**. Alternatively, click the **Edit** button in the **Step Properties** area of the **General** tab.
2. In the **Edit Step** dialog box, change the definition information for the step, and then click the **OK** button.

Figure 3-5: **Edit Step** dialog box

**Edit Step**

Specify step attributes.

Step ID: \* osReadCSVExcelFileColumn

Step Name: \* Acquisition of column data from a CSV/Excel file

Description:

**Component**

**Acquisition of column data from a CSV/Excel file** [Copy] [View]

Acquires the data in the specified column from a CSV or Excel file.

**Vendor Name:** Hitachi,Ltd.

**Version:** 02.00.01

**Key Name:** osReadCSVExcelFileColumn

**Vendor ID:** com.hitachi.software.dna.cts.jp1

**Tags:** Gather OS information, Linux, Windows

**Registered:** 2016-11-28 14:56:33

**Last Updated:** 2016-11-28 14:56:33

**Next Step Conditions**

**Condition:** Determine by Threshold

**Error Threshold:** \* 0

**Use Warnings:** ☐

**Warning Threshold:**


\* Required [OK] [Cancel]

## Operation result

The edited information for the step is set.



### Tip

You can also change the definition information for a step by clicking the step in the **Flow** area, and then clicking the wrench icon (  ) above the step.

---

## Related topics

- [3.4.3 Settings in step definition information](#)
- 

### 3.4.3 Settings in step definition information

The definition information that can be set by users for a step are the step ID, step name, description, and subsequent-step execution condition. Set these items in the **Create Step** dialog box or **Edit Step** dialog box.

Table 3-2: Items displayed in the **Create Step** dialog box or **Edit Step** dialog box

Item		Description
Step	Step ID	A value that uniquely identifies the step. For the step ID, specify a value that is unique within the flow (not including the hierarchy flow). By default, this field displays the component ID. If there is more than one step with the same step ID in a given flow, <code>_n</code> is appended to the end of the step ID (where <code>n</code> is a unique integer starting from 2), which is displayed in the format <code>step-ID_n</code> . Note that if <code>step-ID_n</code> is longer than 30 characters, the excess characters are truncated at the end of the step ID, and the shortened step ID is displayed with the <code>_n</code> suffix.
	Step Name	The value you specify in this field appears with the icon for the step in the <b>Flow</b> area. More than one step can have the same name within a given flow. If there is more than one step with the same name, <code>_n</code> is appended to the end of the step name (where <code>n</code> is a unique integer starting from 2), which is displayed in the format <code>step-name_n</code> . Note that if <code>step-name_n</code> is longer than 64 characters, the excess characters are truncated at the end of the step name, and the shortened step name is displayed with the <code>_n</code> suffix.
	Description	The description you specify in this field is displayed in the <b>Flow</b> area. This field is blank by default.
Component		Displays information about the component.
Next Step Conditions		Set whether to execute the subsequent step according to the return value of the plug-in executed by the current step.

---

## Related topics

- [3.4.4 Overview of subsequent step conditions](#)
- 

### 3.4.4 Overview of subsequent step conditions

You can set whether to execute a subsequent step based on the return value of the previous step.

In most situations, the return value of the step is the same as the return value of the plug-in. For details about the relationship between the return values of steps and plug-ins, see [6.4.10 Relationship of command and script return values to the return values of plug-ins and steps](#).

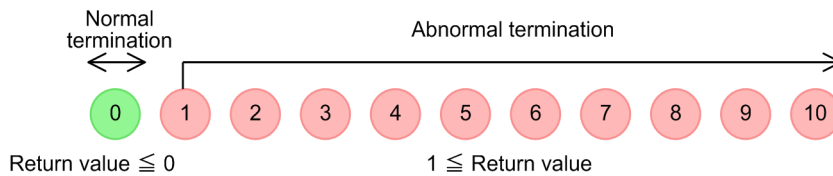
Note that the return value of the step that uses a service component is always 0.

#### Types of subsequent step conditions:

##### Determine the return value based on the threshold

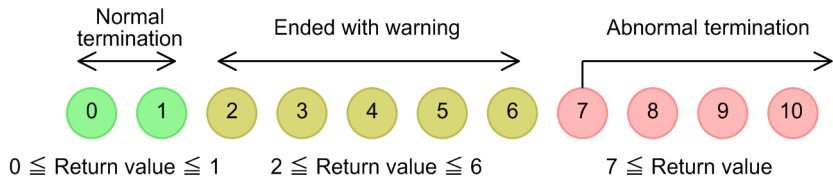
The following figures give examples of setting judgment values and warning values.

Figure 3-6: When judgment value is 0 and no warning value is set (default)



- If the return value is equal to or less than the judgment value, the subsequent step is executed (normal termination).
- If the return value is greater than the judgment value, the subsequent step is not executed (abnormal termination).

Figure 3-7: When judgment value is 6 and warning value is 2



- If the return value is equal to or less than the judgment value, the subsequent step is executed (normal termination). However, if the return value is equal to or greater than the warning value and equal to or less than the judgment value, the subsequent step is executed and the task terminates with a warning. In this case, the task status appears as **Failed** (after the task has finished) or **In Progress (with Error)** (while the task is still running) to indicate that the warning value was exceeded. In the **Flow** area, the step is recorded as having terminated with a warning. When you specify a warning value, specify a value that is equal to or less than the judgment value. You do not need to specify a warning value.
- If the return value is larger than the judgment value, the subsequent step is not executed (abnormal termination).

#### Always succeed regardless of return value

Subsequent steps are always executed, regardless of the return value of the plug-in.

#### Always fail regardless of return value

The step ends abnormally regardless of the return value of the plug-in. Subsequent steps are not executed.

### Plug-ins for which subsequent step conditions cannot be set

You cannot specify subsequent step conditions for service components and the plug-ins that control the following flows:

- Branch by returncode plug-in
- Test value plug-in
- Interval plug-in
- Abnormal-end plug-in
- Branch by property value plug-in

### 3.4.5 Conditional expressions that use arrows to indicate a connection with subsequent steps

In the **Next Steps** tab, specify a conditional expression that uses arrows to connect the selected step, which was selected in the **Flow** area of the **Flow** tab in the **Service Builder Edit** window, and the subsequent steps.

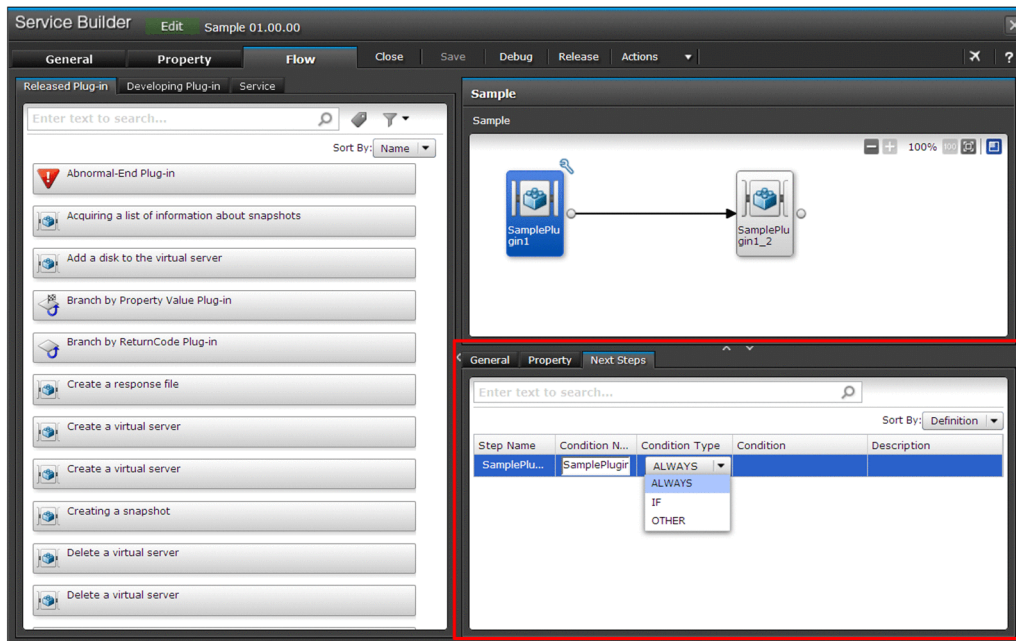


Table 3-3: Items displayed in the **Next Steps** tab

Item	Description
Step Name	The name of a subsequent step is displayed.
Condition Name	The value specified here will be displayed for an icon of a step in the <b>Flow</b> area or in the list of the subsequent steps as the name of a conditional expression using arrows. By default, the name of the subsequent step is displayed.
Condition Type	<p>Select the type of a conditional expression using arrows from the pull-down menu.</p> <ul style="list-style-type: none"> <li>• ALWAYS</li> <li>• IF</li> <li>• OTHER</li> </ul> <p>By default, ALWAYS is displayed. When selecting OTHER, you must specify, in the <b>Next Steps</b> tab, a step for which IF is selected as <b>Condition Type</b> (the type of a conditional expression using arrows).</p>
Condition	The value specified here will be displayed for an icon of a step in the Flow area or in the list of the subsequent steps as a conditional expression using arrows. You can specify this item only when IF is selected as <b>Condition Type</b> (the type of a conditional expression using arrows).
Description	The value specified here will be displayed for an icon of a step in the Flow area or in the list of the subsequent steps as a description of a conditional expression using arrows. You can specify this item only when IF is selected as <b>Condition Type</b> (the type of a conditional expression using arrows).



## 3.5 Defining the execution order of steps

In the **Flow** area of the **Flow** tab in the **Service Builder Edit** window, you can define the execution order of steps by connecting them using relational lines.

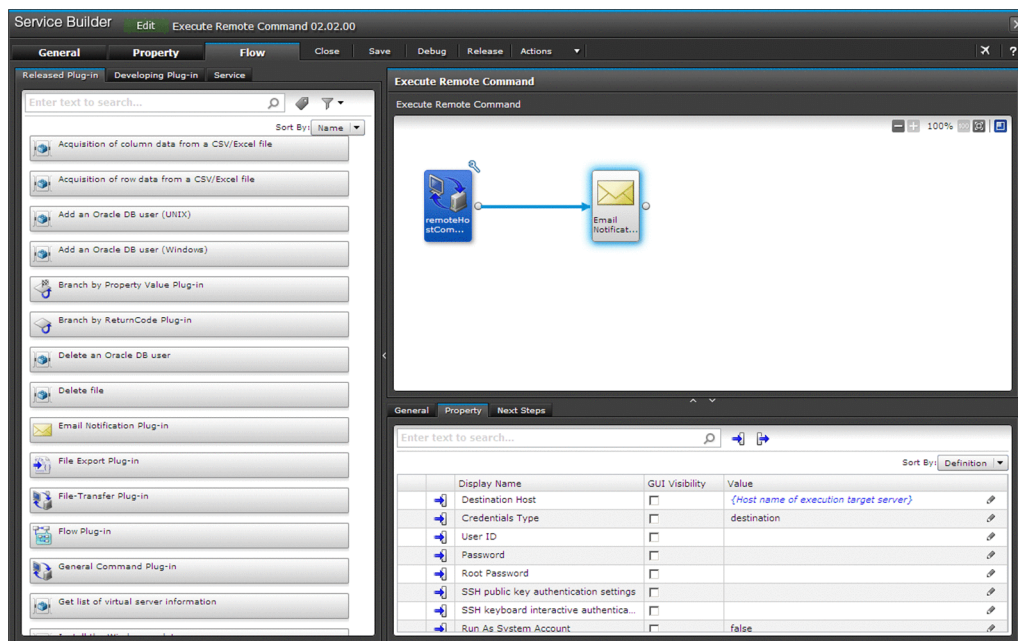
### 3.5.1 Procedure for defining the execution order of steps

After adding steps, the user connects the steps by using relational lines so that the steps will be arranged in the order they are executed in a task. By connecting steps by using relational lines, the execution order of the steps can be defined.

#### To define the execution order of steps:

1. To enable the Auto Complete function for property values, right-click the canvas in the **Flow** area on the **Flow** tab of the **Service Builder Edit** window, and select **Auto-completion of property values**. If there is a tick beside **Auto-completion of property values** in the right-click menu, the Auto Complete function for property values is already enabled.
2. In the **Flow** area, drag the white circle of the icon for the step that will be executed first, to the icon for the step that will be executed next. When the color of the icon changes, release the mouse button.

Figure 3-8: **Service Builder Edit** window **Flow** tab



### Operation result

Steps are connected by relational lines.




#### Related topics

- [3.5.2 Auto-completion of property values](#)
- [3.5.3 Operations that can be performed on steps and relational lines](#)
- [3.5.5 Behavior when relational lines connect to multiple steps](#)
- [3.5.6 Scenarios where relational lines cannot be drawn](#)

- [3.5.7 Drawing relational lines when processing branches](#)

## 3.5.2 Auto-completion of property values

Auto-completion of property values is a function of automatically setting mapping of property values when defining the execution order of steps in the **Flow** area. If you enable auto-completion of property values, property mapping of the connection-source step and connection-destination step is automatically set when steps are connected by a relational line. If property mapping has already been set before auto-completion is performed, the old setting is overwritten with the new setting.

In the **Step Properties** area,  is displayed for the properties for which auto-completion was performed.  remains displayed until another step is selected. If  is displayed for a property, check whether the desired setting has been specified for the step for which auto-completion was performed. If the setting for the step is not what you designed, set the mapping again.

The following describes the method and behavior when setting mapping:

Setting mapping for the input property and the output property of steps

If the output property of the connection-source step and the input property of the connection-destination step of a relational line have the same property key, the reference to the output property is set for the input property.

Setting mapping for the input properties of steps

If the input properties of the connection-source step and the connection-destination step of a relational line have the same property key and data type, the input property value of the connection-source step is set for the input property value of the connection-destination step.

Note that, even if you change the input property value of the connection-source step after auto-completion is performed, the input property value of the connection-destination step will not be changed.

### Tip

Auto-completion is not performed in the following cases:

- The connection-source step and connection-destination step of a relational line are using service components, and mapping has already been set between the internal step properties.

If auto-completion of property values is not performed in other than the above cases, the connection-source step or connection-destination step might be deployed by using any version earlier than JP1/AO 10-02. In that case, deploy the steps again by using the same components. Then these steps will be subject to auto-completion.

## Related topics

- [3.6.1 Overview of step properties](#)
- [3.6.3 Procedure for mapping step property values](#)
- [3.6.4 Overview of property mapping](#)
- [3.6.5 Whether properties can be mapped depending on the visibility or data type](#)
- [4.2.1 Overview of service property](#)



### 3.5.3 Operations that can be performed on steps and relational lines

You can delete, copy, cut, or paste steps and relational lines by right-clicking and selecting the corresponding menu item.

You can select multiple steps and relational lines. In the **Flow** area, drag your mouse and select a range of steps you want to select, or click the steps you want to select while pressing the **Ctrl** key.

The steps and relational lines in the selected range are included in any cut operation. If the selected range contains a hierarchical flow, all subordinate flows are included in any cut operation. Only the data cut by the latest cut operation is cached.

The relational lines that were connected with the steps outside the selected range will be deleted from the **Flow** area immediately after a cut or delete operation.

---

#### Related topics

- [3.5.4 Information inherited when pasting steps or relational lines](#)
- 

### 3.5.4 Information inherited when pasting steps or relational lines

When you paste a step or relational line you have copied or cut, some information associated with the item is inherited at the destination. After pasting a step or relational line, review the definition information for the destination development service template as needed.

The following table shows the information related to steps or relational lines that is inherited when you paste the item.

Table 3-4: Information inherited when pasting steps or relational lines

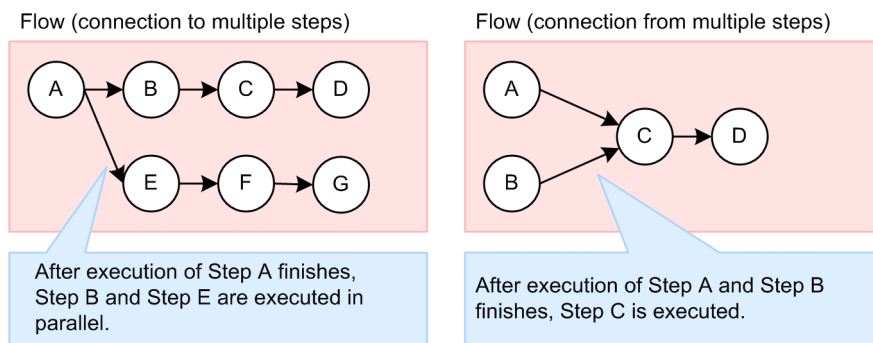
Copied or cut item	Information for copied or cut item	Information at paste destination
Step	Step ID	A step ID must be unique within the flow (not including subordinate hierarchy flows). If there is no step with the same step ID at the paste destination, the information of the copy or cut source is inherited as-is. If there is a step with the same step ID, the step ID of the pasted step changes to <i>step-ID_n</i> , where <i>n</i> is a unique integer of 2 or higher. If <i>step-ID_n</i> is longer than 30 characters, the excess characters are truncated from the end of the step ID.
	Step name	If there is no step with the same name at the paste destination, the information of the copy or cut source is inherited as-is. If there is a step with the same name, the step name of the pasted step is changed to <i>step-name_n</i> , where <i>n</i> is a unique integer of 2 or higher. If <i>step-name_n</i> is longer than 64 characters, the excess characters are truncated from the end of the step ID.
	Description	The information from the copy or cut source is pasted to the destination as-is.
	References to plug-in information	
	Subsequent-step execution conditions	
	Input property settings	

Copied or cut item	Information for copied or cut item	Information at paste destination
Step	Output property settings	The information from the copy or cut source is pasted to the destination as-is.
	Related service properties	If a step uses a development plug-in or release plug-in, the step is added to the property group for the default property. If a step uses a service component, information about the property group is also inherited.
Relational lines	Direction	The system copies information about the direction of relational lines. However, lines that connect to steps outside the selected range are not copied.

### 3.5.5 Behavior when relational lines connect to multiple steps

A flow can contain relational lines that connect one step to several, and several steps to one. When lines are drawn in these ways, the next step is executed only after every connected step has finished executing.

Figure 3-9: Example of connecting multiple steps

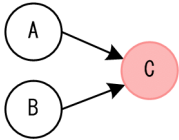
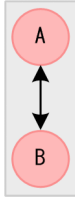
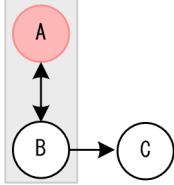
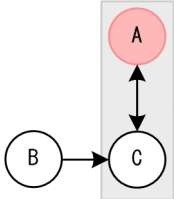




### 3.5.6 Scenarios where relational lines cannot be drawn

You cannot connect relational lines in certain configurations. For example, you cannot draw a relational line that would result in a recurring loop within a flow, or whose source and destination are the same step.

Table 3-5: Restrictions on relational lines

Restricted relational line connection	Details	Example
Relational lines that form a loop	You cannot use relational lines in a way that creates a loop within a flow.	
Relational lines that connect a step to itself	You cannot draw a relational line if the source and destination steps of the line are the same.	
Identical relational lines	You cannot create parallel relational lines with the same source and destination step.	

Restricted relational line connection	Details	Example
Multiple inputs to a branch by returncode plug-in	Because a branch by returncode plug-in judges the return value of the preceding step, you cannot draw relational lines to such a plug-in from multiple steps. Only one step can serve as the preceding step of a branch by returncode plug-in.	
Connecting a branch by returncode plug-in or branch by property value plug-in to a branch by returncode plug-in or branch by property value plug-in	You cannot draw a relational line that connects a branch by returncode plug-in or branch by property value plug-in to a branch by returncode plug-in or branch by property value plug-in. However, you can connect such a plug-in as a succeeding step.	
Relational lines from a branch destination step of a branch by returncode plug-in or branch by property value plug-in	A branch by returncode plug-in or branch by property value plug-in can only have one branch destination step, which cannot be connected further by a relational line. If you need to use several steps, draw a connection line to a flow plug-in.	
Relational lines to a branch destination step of a branch by returncode plug-in or branch by property value plug-in	You cannot draw a relational line to the branch destination step of a branch by returncode plug-in or branch by property value plug-in.	

Legend:  : branch by returncode plug-in or branch by property value plug-in  : Other plug-in

## : Other plug-in

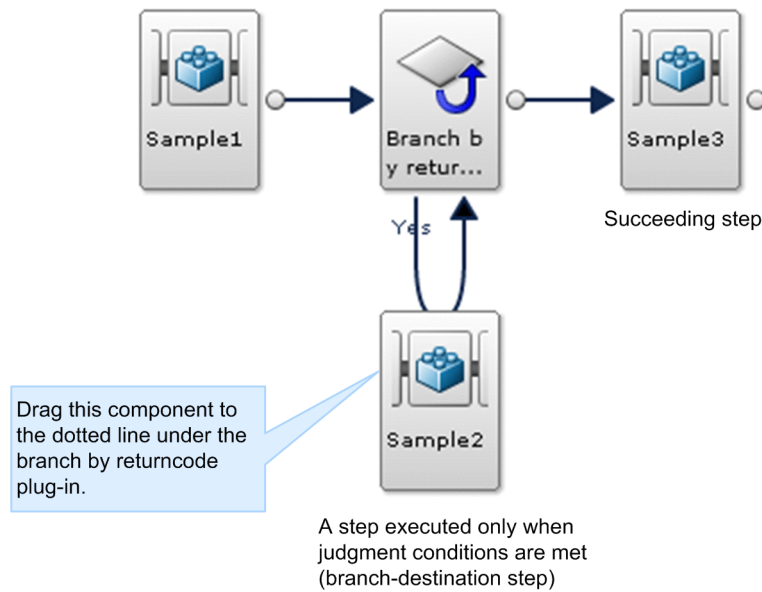
- [3.5.7 Drawing relational lines when processing branches](#)
- *branch by returncode plug-in* in the *JP1/Automatic Operation Service Template Reference*

## 3.5.7 Drawing relational lines when processing branches

If you want to execute a particular step only when judgment conditions are met, you can use a branch by returncode plug-in or branch by property value plug-in. When using such a plug-in, connect the branch-destination component to the plug-in by dragging the branch-destination component to the dotted line under the plug-in.

When the conditions are met, the branch destination step is executed first, followed by the succeeding step. If the conditions are not met, the succeeding step is executed without first executing the branch destination step.

Figure 3-10: Example of relational lines connecting a branch by returncode plug-in



- The connection with the succeeding step is represented by a single-headed arrow.
- The relational line that connects a step executed only when the judgment conditions are met is represented by a double-headed arrow. There cannot be more than one step executed when judgment conditions are met.
- If the judgment conditions are met, the steps are executed in the following order: sample1, branch by returncode plug-in, sample2, and then sample3.
- If the conditions are not met, the steps are executed in the following order: sample1, branch by returncode plug-in, and then sample3.

## 3.6 Setting step properties

---

You can set step properties by directly specifying the values or by mapping other property values.

### 3.6.1 Overview of step properties

A step property is a property defined for a step. Plug-in properties contained in the components used as steps are displayed as step properties in the **Step Properties** area on the **Flow** tab of the **Service Builder Edit** window.

You can set step properties in the following ways:

- Directly specify the value of an input property.
- Inherit the value by mapping the properties of the previous and succeeding steps.
- Elevate a property to a service property to enable the input property value to be specified when the service is set and executed.
- Elevate a property to a service property to enable you to check the output property value in the **Task Details** window.

---

#### Related topics

- [3.6.2 Procedure for directly specifying the input property values of steps](#)
  - [3.6.3 Procedure for mapping step property values](#)
  - [3.6.6 Procedure for elevating step properties to service properties](#)
  - [3.6.7 Example of defining step properties](#)
- 

### 3.6.2 Procedure for directly specifying the input property values of steps

You can directly specify a fixed value in a service template for an input property of a step.

#### To directly specify the input property value of a step:

1. Select a step in the **Flow** area on the **Flow** tab of the **Service Builder Edit** window.
2. On the **Properties** tab of the **Step Properties** area, click the icon for the input property.
3. Select the row corresponding to the property whose value you want to specify directly, and then click the pencil icon.
4. In the **Specify Component Input Property Mapping Parameters** dialog box, select **Direct Input** for Setting Method.

Figure 3-11: **Specify Component Input Property Mapping Parameters** dialog box (when **Direct Input** is selected for Setting Method)

5. In the **Value** text box, enter the value for the step property.



#### Tip

If you want to specify a combination of another property value and any character string, click the **Insert Property** button. In the **Select Reference Property** dialog box, select a property, and then click the **OK** button. Then, the property key for the selected property is inserted in the **Value** text box.

6. Click the **OK** button.

## Operation result

The step property value is set.



#### Tip

You can also specify a step property value in the text box in the **Value** column of the **Step Properties** area. In this case, if the data type of the step property is list, you can select a value from the pull-down menu. If the data type is date, you can select a value from the calendar.

---

## Related topics

- [3.6.7 Example of defining step properties](#)
-

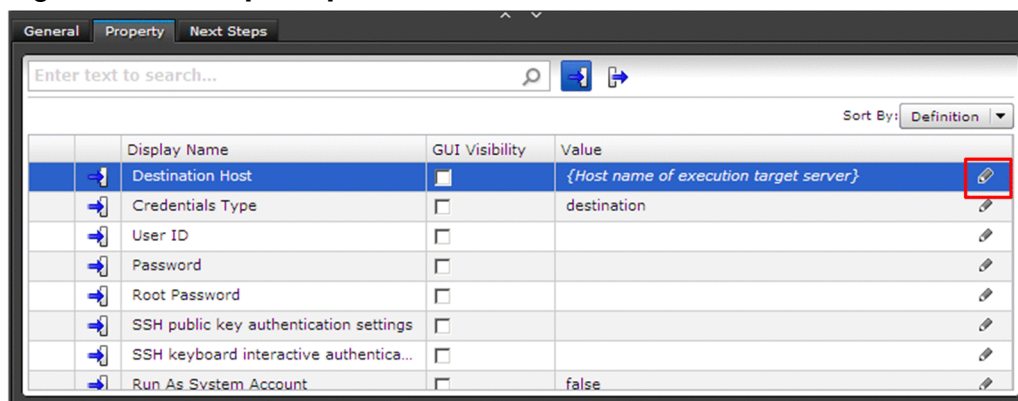
### 3.6.3 Procedure for mapping step property values

If you map properties, the property value can be inherited between the properties. For example, if you map the output property of Step A to the input property of Step B, the value of the output property of Step A can be inherited by the input property of Step B.

#### To set mapping of step properties:

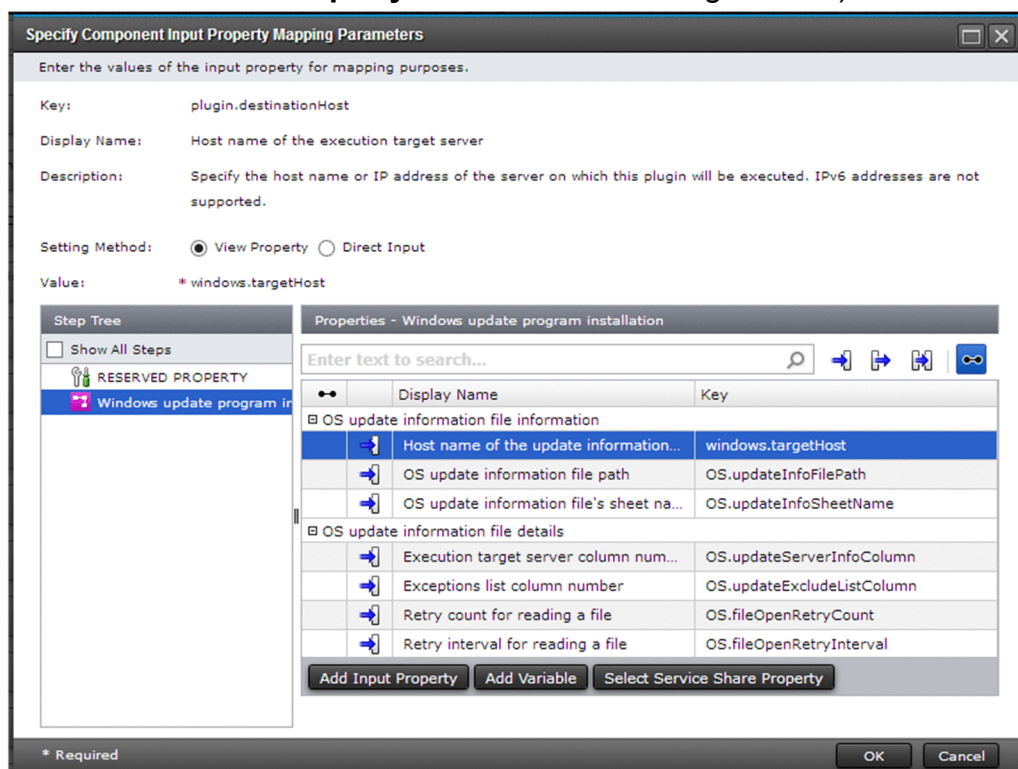
1. Select a step in the **Flow** area on the **Flow** tab of the **Service Builder Edit** window.
2. In the **Step Properties** area, click the **Property** tab and then click the icon for the input or output property.
3. Select the row corresponding to the property you want to map, and then click the pencil icon.

Figure 3-12: **Step Properties** area




4. In the **Specify Component Input Property Mapping Parameters** dialog box or **Specify Component Output Property Mapping Parameters** dialog box, select the **View Property** radio button.  
Only properties that can be mapped are shown in the **Properties** area.

Figure 3-13: **Specify Component Input Property Mapping Parameters** dialog box (when **View Property** is selected for Setting Method)



## Tip

- If you click the Available icon (  ), filtering switches off and the properties that cannot be mapped are also displayed. If you point to an error icon, the reason why the property cannot be mapped is displayed.
- Entering information into the **Specify Component Input Property Mapping Parameters** dialog box
  - To display service properties, reserved properties, or variables  
Select a service template name in the **Step Tree** area. You can display the service properties that belong to a property group by clicking the icon to the left of the property group name. Reserved properties and variables are displayed in the default property group.
  - To display step properties  
In the **Step Tree** area, select a step name. The properties for the step are displayed in the **Properties** area. By default, the steps before the selected step are displayed in the **Step Tree** area. If you select the **Show All Steps** check box, all steps in the flow are displayed in the **Step Tree** area.

5. In the **Properties** area, select the property you want to map.

6. Click the **OK** button.

## Operation result

Properties are mapped.

## Tip

- You can create a new service property and map it to a step property. The service properties you created are displayed in the **Specify Component Input Property Mapping Parameters** dialog box or **Specify Component Output Property Mapping Parameters** dialog box, and can be mapped in the same way as step properties.
- If the visibility or data type is different between the mapping-source property and the mapping-destination property, an error might occur while the service template is built. For details about whether properties can be mapped depending on the visibility or data type, see [3.6.5 Whether properties can be mapped depending on the visibility or data type](#).
- By default, the Message service property is provided for output properties. Message is a service property used to output the execution result to the **Task Details** window. If you map Message to the output property of a step, you can output the output property value as the execution result.  
Note that you can edit or delete Message. If you delete Message and then add a service property with the same property key, the output property value is displayed in the **Task Details** window.

## Related topics

- [3.6.4 Overview of property mapping](#)
- [3.6.7 Example of defining step properties](#)



## 3.6.4 Overview of property mapping

Property mapping is the setting that inherits a value from one step property or service property to another step property or service property. This enables you to dynamically set property values.

### Setting mapping for an input property

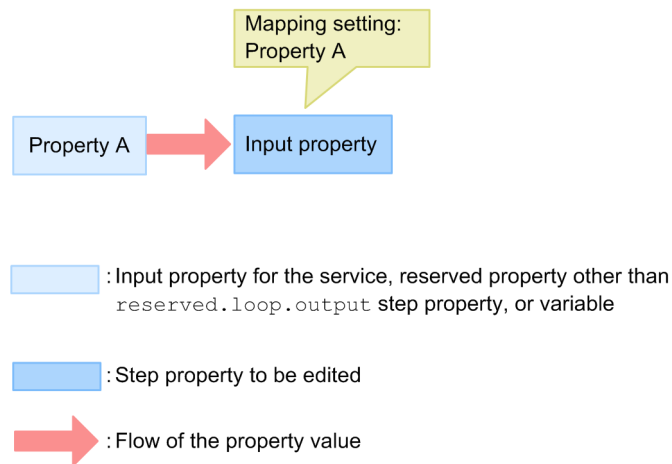
When you edit an input property and set mapping, the value of the property being mapped is inherited by the input property being edited.

You can map the following properties to the input property:

- Step properties
- Input property for the service
- Reserved property other than `reserved.loop.output`
- Variables

The following figure shows an example of data flow when mapping is set for the input property.

Figure 3-14: Data flow when mapping is set for the input property



In this figure, Property A is mapped to the input property. Therefore, the value specified or output to Property A is inherited by the value for the input property.

### Setting mapping for an output property

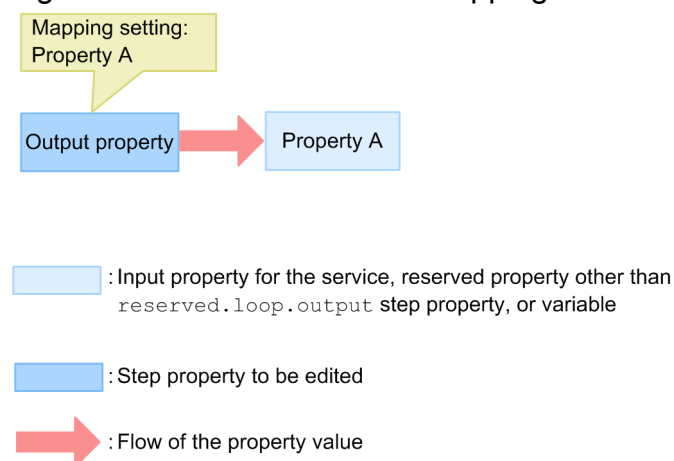
When you edit an output property and set mapping, the value of the output property being edited is inherited by the property being mapped.

You can map the following properties to an output property:

- Output property for the service
- Reserved property `reserved.loop.output`
- Variables

The following figure shows an example of data flow when mapping is set for the output property.

Figure 3-15: Data flow when mapping is set for the output property



In this figure, Property A is mapped to the output property. Therefore, the value output to the output property is inherited by the value for Property A.

### ! Important

If you map the output property for which mapping is being set to an input property, an error occurs while the service template is built. For example, if the value of the output property of a step is set to be inherited by the output property of a service, the value cannot be inherited by the input property of the next step.

## Related topics

- [3.6.3 Procedure for mapping step property values](#)
- [3.6.5 Whether properties can be mapped depending on the visibility or data type](#)

## 3.6.5 Whether properties can be mapped depending on the visibility or data type

We recommend that you use the same visibility and data type for both the property from which a value is inherited and the property that inherits the value when the properties are mapped. If the visibility or data type for the properties are different, an error might occur while the service template is built, depending on the combination of the visibility settings or data type settings. If an error occurs, in the **Flow** tab of the **Service Builder Edit** window, check and (if necessary) change the visibility or data type for the service properties.

### Whether properties can be mapped depending on the visibility

For the property that inherits the property value, specify the same level or a lower level of visibility than the property whose value is inherited. The lower level of visibility means that the range open to the public (the windows that display properties) is limited. If the property that inherits the value has a wider range open to the public than the property whose value is inherited, an error occurs when the service template is built.

The following table describes whether properties can be mapped depending on the combination of visibility settings.

Table 3-6: Whether properties can be mapped depending on the combination of the visibility settings when input properties are edited

Visibility for the property whose value is inherited		Visibility for the property that inherits the value (input property)		
		Input property of the service (with a related step)		Input property of the step <sup>#</sup>
		Edit and Submit Window	Edit Window Only	
Input property of the service (with no related steps)	Edit and Submit Window	Y	Y	Y
	Edit Window Only	N	Y	Y
Input property of the service (with a related step)	Edit and Submit Window	Y	Y	Y
	Edit Window Only	N	Y	Y
Variable <sup>#</sup>		N	N	Y
Input property of the step <sup>#</sup>		N	N	Y
Output property of the step <sup>#</sup>		N	N	Y

Legend:

Y: The properties can be normally mapped. N: The properties can be mapped, but an error occurs when the service template is built.

#

Step properties and variables are internal properties that can be viewed or set when users in a Develop or higher role develop service templates. Therefore, the visibility is at a lower level than a service property.

Table 3-7: Whether properties can be mapped depending on the combination of the visibility settings when output properties are edited

Visibility for the property whose value is inherited (output property)	Visibility for the property that inherits the value		
	Output property of the service (with no related steps)	Output property of the service (with a related step)	Variable <sup>#</sup>
Output property of the service (with a related step)	Y	N	N
Output property of the step <sup>#</sup>	Y	N	Y

Legend:

Y: The properties can be normally mapped. N: The properties can be mapped, but an error occurs when the service template is built.

#

Step properties and variables are internal properties that can be viewed or set when users in a Develop or higher role develop service templates. Therefore, the visibility is at a lower level than a service property.

## Whether properties can be mapped depending on the data type

For the property that inherits the property value, specify a data type that can handle the data type of the property whose value is inherited. Note that step properties and the reserved.loop.output reserved property have no data type. However, if the component that the step is based on is a basic plug-in, data type is set for some of the step properties. If such a

step property is elevated to a service property, you can check the data type in the **Edit Input Property for Service** dialog box.

The following table describes whether the properties can be mapped depending on the combination of the data type settings.

**Table 3-8: Whether properties can be mapped depending on the combination of the data types when input properties are edited**

Data type for the property whose value is inherited		Data type for the property that inherits the value (input property)								
Property type	Data type	boolean	integer	string	double	date	password	list	composite	No data type
Input property	boolean	Y	N	Y	N	N	N	N	Y	Y
	integer	N	Y	Y	N	N	N	N	Y	Y
	string	N	N	Y	N	N	N	N	Y	Y
	double	N	N	Y	Y	N	N	N	Y	Y
	date	N	N	Y	N	Y	N	N	Y	Y
	password	N	N	N	N	N	Y	N	N	Y
	list	N	N	Y	N	N	N	Y	Y	Y
	composite	N	N	N	N	N	N	N	Y	Y
	No data type	N	N	Y	N	N	N	N	Y	Y
Output property	string	Y	Y	Y	Y	N	N	Y	Y	Y
	password	N	N	N	N	N	Y	N	N	Y
	composite	N	N	N	N	N	N	N	Y	Y

Legend:

Y: The properties can be normally mapped. N: The properties can be mapped, but an error occurs when the service template is built.

**Table 3-9: Whether properties can be mapped depending on the combination of the data types when output properties are edited**

Data type for the property whose value is inherited (output property)	Data type for the property that inherits the value			
	string	password	composite	No data type
string	Y	N	Y	Y
password	N	Y	N	N
composite	N	N	Y	N
No data type	Y	Y	Y	Y

Legend:

Y: The properties can be normally mapped. N: The properties can be mapped, but an error occurs when the service template is built.

## Tip

When properties are mapped, the restrictions on input characters and maximum length that are set for the property whose value is inherited are ignored, and the restrictions on the property that inherits the value are applied. Therefore, set restrictions on the property that inherits the value by considering the property value that is inherited.

## Related topics

- [3.6.4 Overview of property mapping](#)
- [3.6.3 Procedure for mapping step property values](#)
- [3.6.6 Procedure for elevating step properties to service properties](#)
- [4.2.1 Overview of service property](#)
- [4.2.4 Items set for input properties of services](#)
- [4.2.5 Items set for output properties of services](#)

## 3.6.6 Procedure for elevating step properties to service properties

If you elevate a step property to a service property, the service property is displayed in the **Property** tab of the **Service Builder Edit** window, and you will be able to edit the service property. For details about service properties, see [4.2.1 Overview of service property](#).

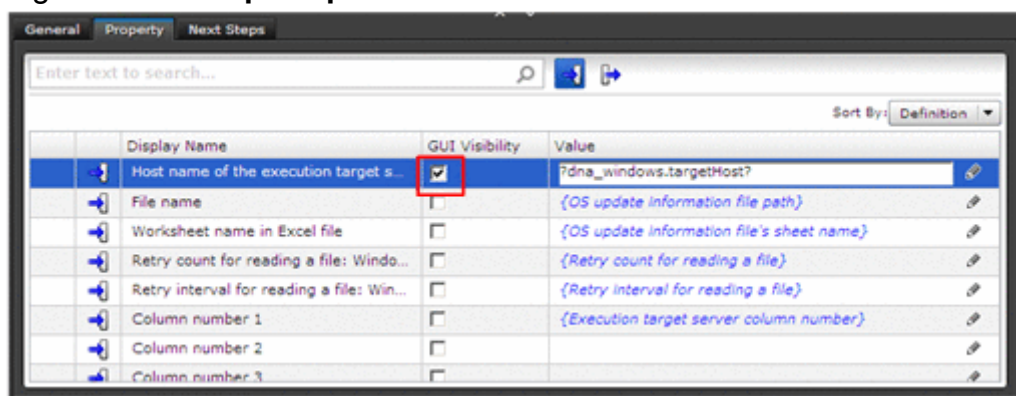
If you elevate the input property of a step to a service property, the service property is displayed in the **Service Definition** window and **Submit Service** window, and you will be able to enter a value when creating or submitting a service.

If you elevate the output property of a step to a service property, after the service is submitted, you will be able to check the value of the output property in the **Output** area of the **Task Details** window.

### To elevate a step property to a service property:

1. In the **Flow** area of the **Flow** tab of the **Service Builder Edit** window, select a step.
2. In the **Step Properties** area, click the icon for input or output properties, and select the check box in the **GUI Visibility** column for the step property you want to elevate to a service property.

Figure 3-16: **Step Properties** area





## Tip

If you select a step that uses a service component, you cannot change the selection of the check box in the **GUI Visibility**.

For a step that uses one of the following components, the check box is not displayed in the **GUI Visibility** because the property of such a step cannot be elevated to a service property.

- Flow plug-in
- Interval plug-in
- Branch by returncode plug-in
- Abnormal-end plug-in

## Operation result

A step property is elevated to a service property, and displayed in the **Properties** tab of the **Service Builder Edit** window.

## Related topics

- [3.6.7 Example of defining step properties](#)

## 3.6.7 Example of defining step properties

This subsection describes an example of defining step properties when creating a service template.

Assume a service template that executes Step A and Step B in this order. Step B is executed for the folder in which the output result of Step A is stored. Values of some properties are not fixed in the service template, but are specified each time a service is executed.

## Definitions

The following table describes the definitions of the step properties.

Table 3-10: Definitions of Step A

Property type	Property name	Definition
Input property	Capacity	The value is fixed at 10. You do not need to change this value.
	Execution server	Specify the value when a service is executed.
Output property	Path to the folder storing the output result	You do not need to check the output result in the <b>Task Details</b> window.

Table 3-11: Definitions of Step B

Property type	Property name	Definition
Input property	Path to the execution folder	Specify the folder to store the output result of Step A.
Output property	Output result of Step B	Check the output result in the <b>Task Details</b> window.

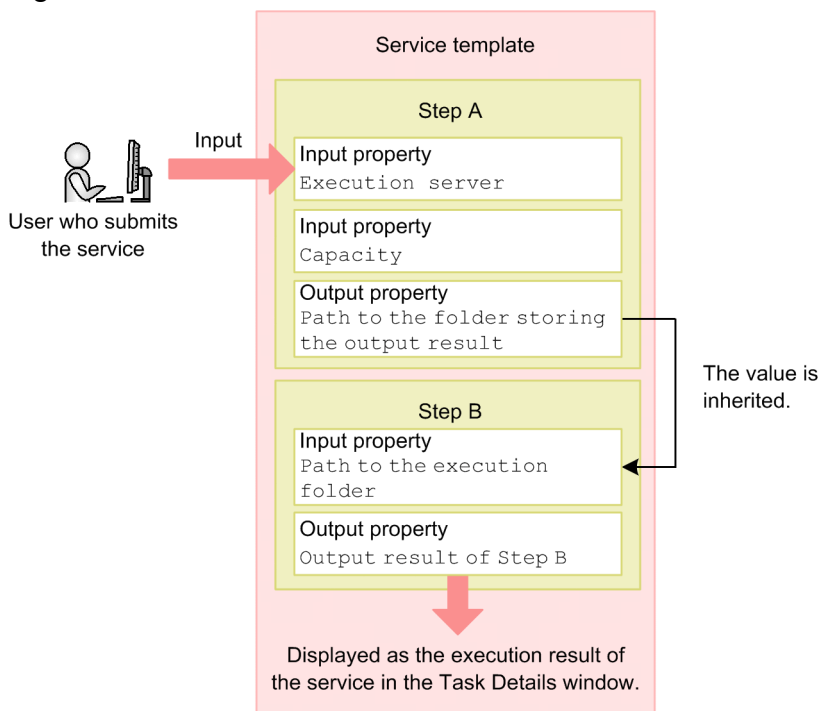
## Settings

- Because the value of the Capacity property is fixed at 10, in the **Step Properties** area on the **Flow** tab of the **Service Builder Edit** window, directly enter 10 in the text box in the **Value** column.
- To specify the value of the Execution server property when a service is executed, in the **Step Properties** area on the **Flow** tab of the **Service Builder Edit** window, select the check box in the **GUI Visibility** to elevate the property to a service property.
- To execute Step B for the folder storing the output result of Step A, select the "Path to the execution folder" property of Step B, and in the **Specify Component Input Property Mapping Parameters** dialog box, specify the "Path to the folder storing the output result" property of Step A.
- After the service is executed, to check the "Output result of Step B" property in the **Task Details** window, in the **Step Properties** area on the **Flow** tab of the **Service Builder Edit** window, select the check box in the **GUI Visibility** to elevate the property to a service property.

## Data flow

The following figure shows the data flow of individual properties.

Figure 3-17: Data flow



1. The value specified by the user who submits the service in the **Submit Service** window is stored as the value of the "Execution server" property of Step A.
2. The value output to the "Path to the folder storing the output result" property of Step A is stored as the value of the "Path to the execution folder" property of Step B.
3. The "Output result of Step B" property of Step B is displayed in the **Task Details** window.

## Related topics

- [3.6.2 Procedure for directly specifying the input property values of steps](#)
- [3.6.3 Procedure for mapping step property values](#)

- 3.6.6 Procedure for elevating step properties to service properties

### 3.6.8 List of reserved properties

A reserved property is a special service property whose property key has a specific definition or purpose in JP1/AO. The property key of a reserved property begins with "reserved". You can use reserved properties by mapping them to step properties in the **Specify Component Input Property Mapping Parameters** dialog box or **Specify Component Output Property Mapping Parameters** dialog box. Users do not need to define or assign values to reserved properties.

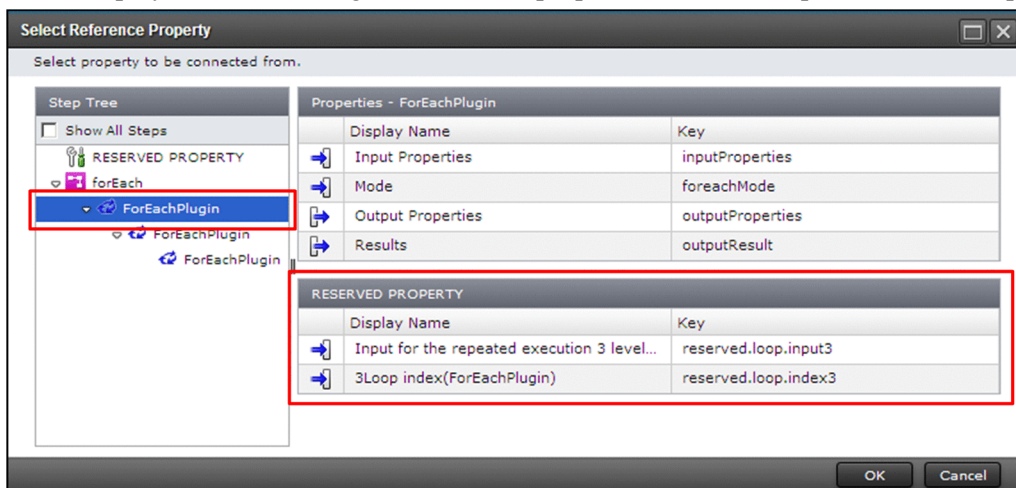
When you map a reserved property to an input property, the value of the reserved property is assigned to a plug-in property when the plug-in is executed.

To use a reserved property, select the **View Property** radio button in the **Specify Component Input Property Mapping Parameters** dialog box.

Alternatively, select the **Direct Input** option, and then click the **Insert Property** button. If you select a reserved property in the **Select Reference Property** dialog box<sup>#</sup>, you can specify (in the **Value** text box) 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.

#

In the **Step Tree** area, if you select a step of a repeated execution plug-in, the **RESERVED PROPERTY** dialog box is displayed. In this dialog box, reserved properties related to repeated execution plug-ins are displayed.



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** radio button 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.

Table 3-12: List of reserved properties

Reserved property key	Description
reserved.loop.index <sup>#1</sup>	<p>References a numerical value from 1 to 99 that indicates how many times a repeated execution plug-in located one level above the selected plug-in has repeated.</p> <p>Of the comma-delimited values specified in the inputProperties property of the repeated execution plug-in, this property stores the position of the parameter to which the current execution applies. You can also reference this value when parallel is set as the execution method of the repeated execution plug-in. To reference this reserved property, specify the property key in the format ?dna_reserved.loop.index?. Like the reserved.loop.input property, you can use this property in any plug-</p>

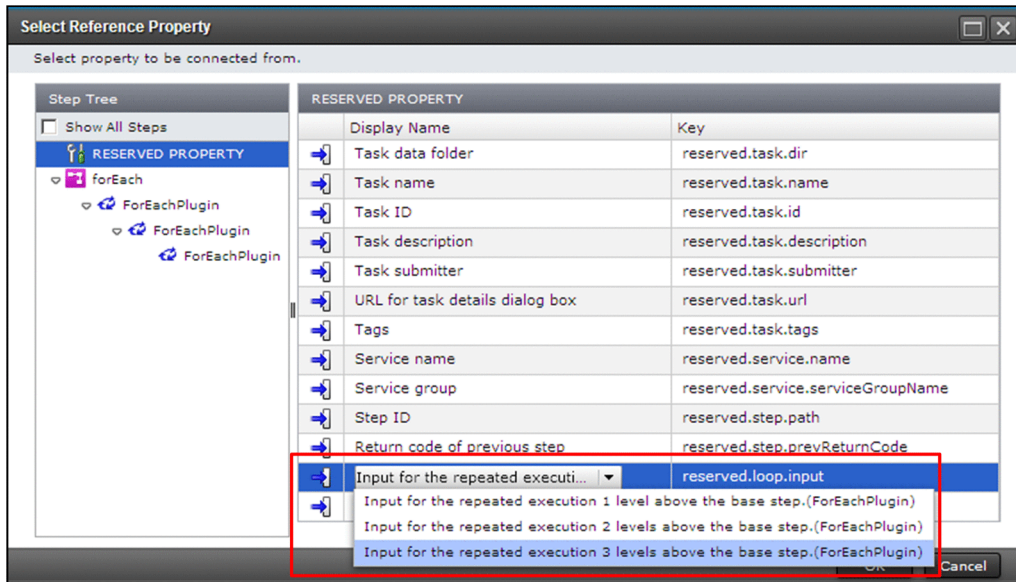


Reserved property key	Description
reserved.loop.index <sup>#1</sup>	in within the flow of the repeated execution plug-in, provided that service properties can be mapped to the plug-in.
reserved.loop.index <sup>N#1</sup>	<p>References a numerical value from 1 to 99 that indicates the number of times that a repeated execution plug-in located <i>N</i> levels above the selected plug-in has been executed.</p> <p>Of the comma-delimited values specified in the inputProperties property of the repeated execution plug-in, this property stores the position of the parameter to which the current execution applies. You can also reference this value when parallel is set as the execution method of the repeated execution plug-in. To reference this reserved property, specify the property key in the format ?dna_reserved.loop.index<sup>N</sup>?. Like the reserved.loop.input property, you can use this property in any plug-in within the flow of the repeated execution plug-in, provided that service properties can be mapped to the plug-in.</p> <p>Note: Above, <i>N</i> represents the integer 2 or 3.</p>
reserved.loop.input <sup>#1</sup>	<p>References the value of the inputProperties input property of a repeated execution plug-in located one level above the selected plug-in.</p> <p>As one 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 executed a maximum of 99 times. To reference this reserved property, specify the property key in the format ?dna_reserved.loop.input?.</p>
reserved.loop.input <sup>N#1</sup>	<p>References the value of the inputProperties property of a repeated execution plug-in that is located <i>N</i> levels above the selected plug-in.</p> <p>As one 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 executed a maximum of 99 times. To reference this reserved property, specify the property key in the format ?dna_reserved.loop.input<sup>N</sup>?.</p> <p>Note: Above, <i>N</i> represents the integer 2 or 3.</p>
reserved.loop.output	<p>Passes values to the outputProperties output property of a repeated execution plug-in.</p> <p>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.</p>
reserved.service.name	<p>References the name of the service from which a task was generated.</p> <p>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.</p>
reserved.step.path	<p>References the ID of the step that is currently being executed.</p> <p>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 displayed 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.</p>
reserved.service.serviceGroupName	<p>References the service group in which the service from which a task was generated is registered.</p> <p>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.</p>
reserved.step.prevReturnCode	<p>Supplies the return value of the preceding step (the step that is the origin of the relational line connected to the plug-in).</p> <p>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.</p>

Reserved property key	Description
reserved.step.prevReturnCode	If you retry a task from a step that references this reserved property without executing the preceding step, the return value from the last time the preceding step was executed is set in this reserved property as the return value of the preceding step. <sup>#</sup>
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 provides a unique folder path at the execution of each task. The folder referenced by this property is created on the JP1/AO server when the task is executed, and deleted when the task is archived. Note that files and folders that start with "task" are reserved in JP1/AO, 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	Supplies the tags set for the task. To reference this reserved property, specify the property key in the format ?dna_reserved.task.tags?. You can use this property in any plug-in to which service properties can be mapped.
reserved.task.url	Supplies the URL for accessing the <b>Task Details</b> window. 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.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 administrator 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.

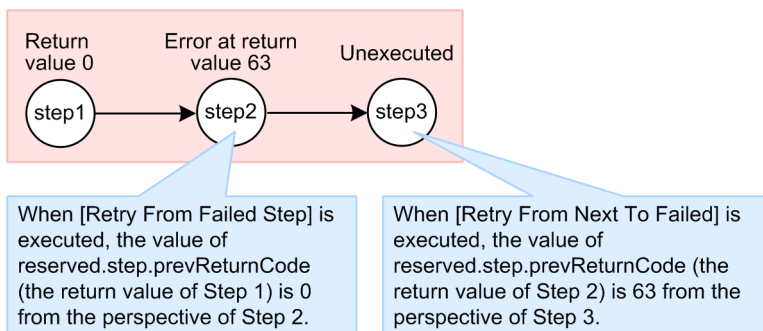
#1

If the definition of repeated execution plug-ins is nested, the input value of a repeated execution plug-in (reserved.loop.input, reserved.loop.input $N$ ) and a value that indicates how many times the repeated execution plug-in has been repeated (reserved.loop.index, reserved.loop.index $N$ ) are displayed together in a pull-down menu. In this case, the step names of repeated execution plug-ins are displayed in parentheses. The following figure shows the details.



#2

The following shows examples of the values assigned to the reserved property when a task is retried:



The illustrated flow consists of Step 1, Step 2, and Step 3. The reserved.step.prevReturnCode property is defined for Step 2 and Step 3. The return value of Step 1 is 0, and the return value of Step 2 is 63, indicating an error. The task fails with Step 3 in Unexecuted status.

In this scenario, if you use the **Retry the Task From the Failed Step** option, the value of reserved.step.prevReturnCode (the return value of Step 1) is 0 from the perspective of Step 2. If you use the **Retry the Task From the Step After the Failed Step** option, the value of reserved.step.prevReturnCode (the return value of Step 2) is 63 from the perspective of Step 3.

### 3.6.9 Warning icon displayed for steps

If a mandatory step property value is omitted, a warning icon is displayed for the step.

Figure 3-18: Example of steps with warning icon displayed



The icon is updated when you edit the value of a step property.

When you copy and paste a step with a warning icon, the pasted step retains the warning icon.

The warning icon is also displayed in the **Step Properties** area.

## 3.7 Managing the versions of components used as steps

For a component used as a step in a service template, you can update the version to the latest or change it to any version.

### 3.7.1 Overview of managing the versions of components used as steps

In JP1/AO, you can change the versions of components used as steps in a development service template in a batch or separately. When you change the versions of components, you can inherit the information set by users (such as property values). You can also check the versions of components used as steps. These functions can be used for components that have multiple versions.

If you change the versions of components using service components that contain further inside service components, the versions of those inside service components are not changed.

### Functions of managing the versions of components used as steps

You can choose **Apply to All** or **Individual apply** as the method of checking and changing the versions of components used as steps. The following notes describe their functions.

#### Apply to All

This function changes the versions of all components used as steps to the latest versions of release plug-ins or service components in a batch. However, the steps that use the latest versions of components are not changed. You can also check the versions of all components used as steps.

#### Individual apply

This function changes the version of a specified component used as a step to any version you want. You can also check the version of a specified component.

The following table describes the difference between **Apply to All** and **Individual apply**.

Table 3-13: Difference between the **Apply to All** and **Individual apply** functions

What is compared	Apply to All	Individual apply
Components whose version is subject to change	Components whose versions are older than the latest.	The specified component (of any versions).
Components after changing the versions	The latest versions of release plug-ins and service components.	The specified version of component.



#### Tip

These functions can be used for development service templates. If you want to change the versions of components used by steps in release service templates, copy the corresponding service templates beforehand, and then use them as development service templates.

### Related topics

- [3.7.2 Procedure for checking the versions of components used as steps](#)

- [3.7.3 Procedure for batch-updating components used as steps to the latest versions](#)
  - [3.7.4 Procedure for changing the version of a component used as a step to any specified version](#)
  - [3.7.5 Information inherited when the versions of components are changed](#)
- 

## 3.7.2 Procedure for checking the versions of components used as steps

You can check the versions of components used as steps.

To check the versions of all components used as steps in a batch, select the **Apply to All** tab. To check the version of a specified component, select the **Individual apply** tab.

### To check the versions on the Apply to All tab

1. In the **Service Builder Home** window, click the **Developing** tab, select the service template whose component versions you want to check, and then click the **Edit** button.
2. In the **Actions** pull-down menu, select **Component Version Management**.
3. In the **Component Version Management** dialog box, select the **Apply to All** tab.
4. In **Current Version of Step List**, you can check the versions of all components used as steps in the service template. For steps whose **Status** is displayed as **Not applied**, you can change the versions of the components to the latest versions of release plug-ins or service components. The steps whose status is displayed as **Applied** already use the latest versions of release plug-ins or components. The steps whose status is displayed as **Not applicable** do not have release plug-ins or service components that can be specified for the change-destination.

### To check the version on the Individual apply tab

1. In the **Service Builder Home** window, click the **Developing** tab, select the service template whose component versions you want to check, and then click the **Edit** button.
2. In the **Actions** pull-down menu, select **Component Version Management**.
3. In the **Component Version Management** dialog box, select the **Individual apply** tab.
4. From the **Component List**, select the component of which you want to check the version.
5. In **Current Version of Step List**, you can check the version of the component used by the selected step. For a step whose **Status** is displayed as **Not applied**, you can change the version of the component. The steps whose status is displayed as **Applied** already use the specified versions of components.

---

### Related topics

- [3.7.1 Overview of managing the versions of components used as steps](#)
- 

## 3.7.3 Procedure for batch-updating components used as steps to the latest versions

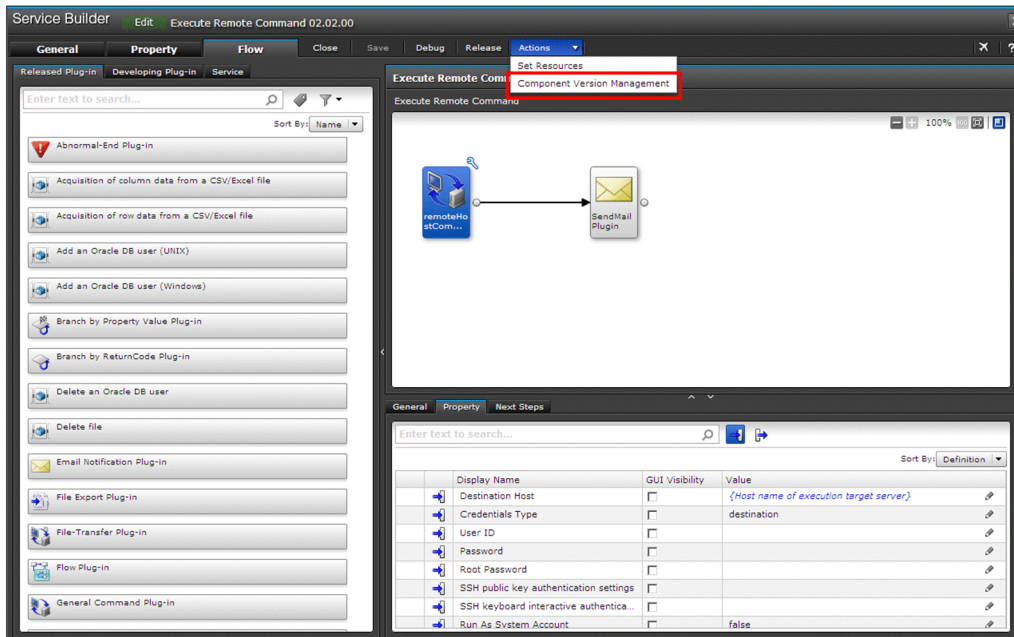
You can change the versions of components used as steps to the latest versions of release plug-ins or service components in a batch.

1. In the **Service Builder Home** window, click the **Developing** tab, select the service template whose component versions you want to check, and then click the **Edit** button.



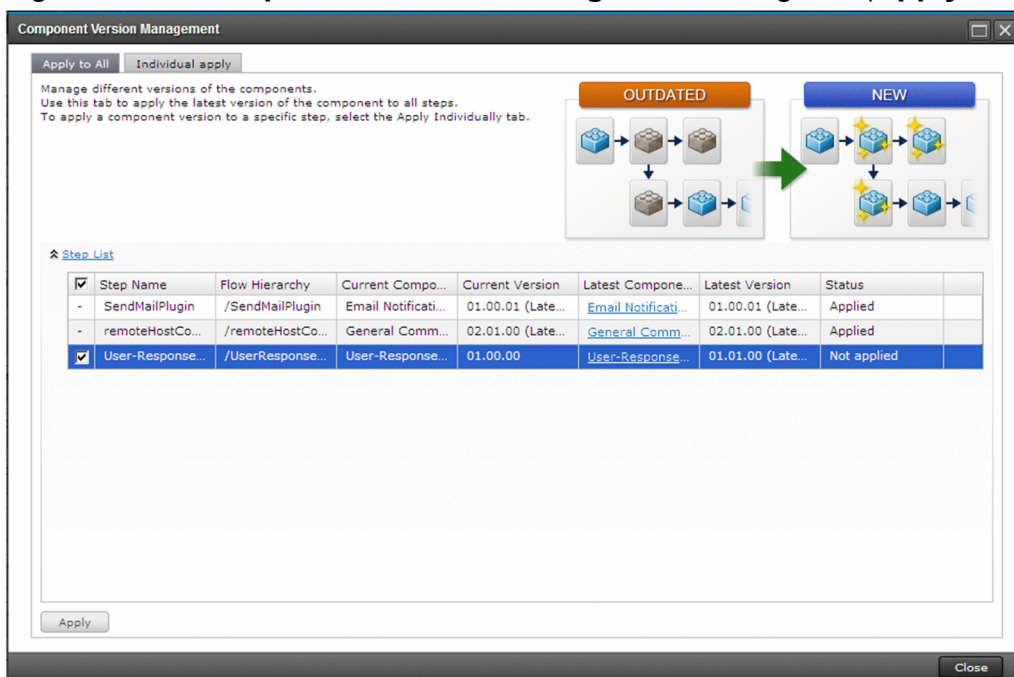
2. From the **Actions** pull-down menu, select **Component Version Management**.

Figure 3-19: Selecting **Component Version Management**



3. In the **Component Version Management** dialog box, select the **Apply to All** tab.

Figure 3-20: **Component Version Management** dialog box (**Apply to All** tab)




### Tip

- If you do not want to update the components used by some steps to the latest versions, clear the check boxes for selecting such steps.
- For the steps in which newer versions of development plug-ins than the latest versions of release plug-ins are used, the check boxes for selecting the steps as the change targets are not selected.

Therefore, to include such plug-ins as the change targets, select the check boxes for selecting the steps.

4. Click the **Apply** button.

All components used by the steps selected in **Step List** are changed to the latest versions of release plug-ins or service components in a batch.

If the property values before version change are not inherited as the properties after version change,  is displayed for such properties. If this icon is displayed, check and (if necessary) change the definition information of the corresponding step. A property value before change is not inherited in the following cases:

- The property has been deleted.
- The value of the property key has been changed.
- The type of the property (input or output) has been changed.

## Operation result

The component used by the selected step is changed to the latest version of release plug-in or service component.

---

### Related topics

- [3.7.1 Overview of managing the versions of components used as steps](#)
  - [3.7.5 Information inherited when the versions of components are changed](#)
- 

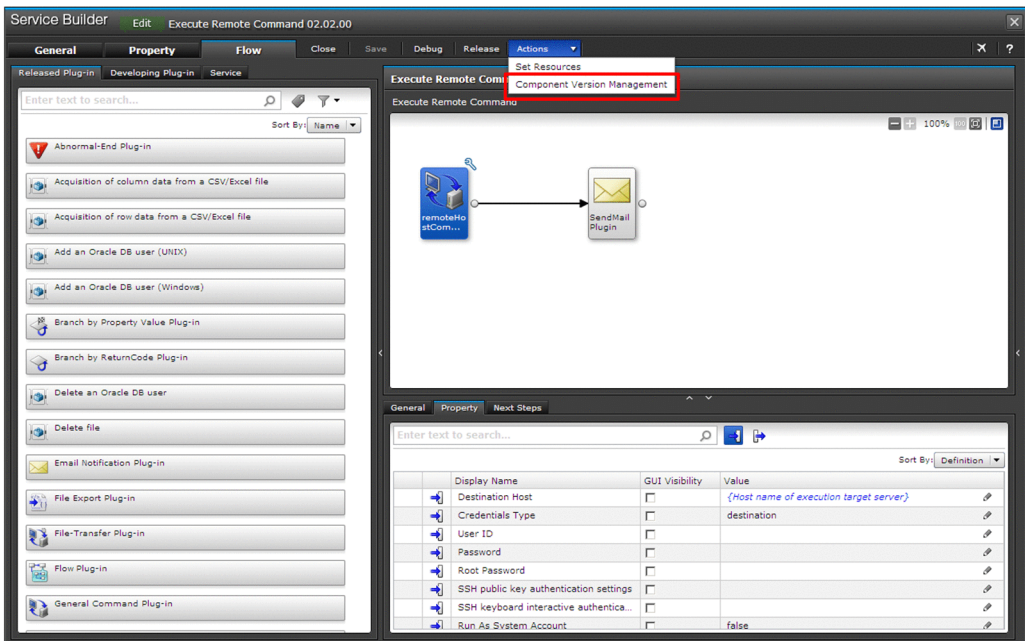
## 3.7.4 Procedure for changing the version of a component used as a step to any specified version

You can change the version of each component used as a step to a specified version.

1. In the **Service Builder Home** window, click the **Developing** tab, select the service template whose component versions you want to check, and then click the **Edit** button.
2. In the **Actions** pull-down menu, select **Component Version Management**.

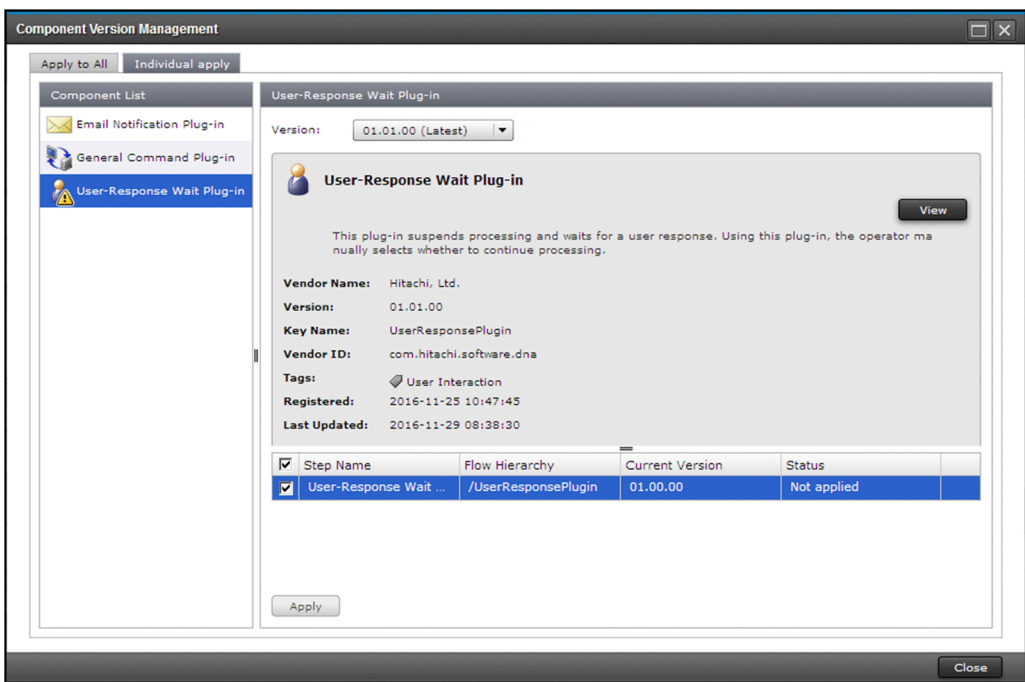



Figure 3-21: Selecting **Component Version Management**



3. In the **Component Version Management** dialog box, select the **Individual apply** tab.

Figure 3-22: **Component Version Management** dialog box (**Individual apply** tab)




4. From **Component List**, select the component you want to change the version.  
In **Component List**, the types of components used as steps are displayed. The types of components are displayed when there are multiple versions of components. If a type of components contains components whose versions are not the latest,  is displayed for that type.
5. To view detailed information about the destination version, select the version from the **Version** pull-down menu, and check the displayed plug-in information. If a **View** button is displayed for a plug-in, you can view detailed information about the plug-in in the **View Plug-in Details** dialog box that appears when you click the **View** button.

Check **Status** of **Step List** for each version you specified. For the steps whose **Status** is displayed as **Not applied**, you can change the versions. For the steps whose status is displayed as **Applied**, you cannot change the versions because the current versions are the same as the versions specified for the destination versions.

6. From the **Version** pull-down menu, select a version you want to specify for the change destination.
7. Among the steps whose **Status** is **Not applied**, select a step for which you want to change the version.
8. Click the **Apply** button.

The step selected in **Step List** is changed to the specified version of the component.

If the properties before version change are not inherited as the properties after version change,  is displayed for such properties. If this icon is displayed, check and (if necessary) change the definition information of the corresponding step. A property value before change is not inherited in the following cases:

- The property has been deleted.
- The value of the property key has been changed.
- The type of the property (input or output) has been changed.

## Operation result

The component used by the selected step is changed to the specified version of the component.

---

### Related topics

- [3.7.1 Overview of managing the versions of components used as steps](#)
  - [3.7.5 Information inherited when the versions of components are changed](#)
- 

## 3.7.5 Information inherited when the versions of components are changed

When you change the versions of components, some information is inherited by the same properties before and after version change.

### Property information inherited when the versions of components are changed

If the property key and type (input or output) of a property are the same before and after "Apply to All" or "Individual apply" is performed, that property is handled as the same property before and after the version change. If a property is added after the version change, the default value is specified for the property. If a property is deleted after the version change, the corresponding property itself is deleted.

The following information is inherited when a property is handled as the same property before and after "Apply to All" or "Individual apply" is performed:

- Property value (including Input Method of the property)
- Property name
- Description
- Display settings
- Default value<sup>#</sup>

#: The default value is updated by the information after version change only when one of the following conditions is applied in a step that uses a service component:

- A property before or after version change references a property in a service component.
- The data type of a property is changed from the composite type to other than the composite type.
- The data type of a property is changed from other than the composite type to the composite type.

Note that, in this case, the default value becomes blank.

## **Property group information inherited when the versions of components are changed**

If the property group key of a property group is the same between before and after "Apply to All" or "Individual apply" is performed, that property group is handled as the same property group before and after the version change. The property groups that exist only in the components after the version change are newly added. The property groups that exist only in the components before the version change are deleted.

The following information is inherited when a property group is handled as the same property group before and after "Apply to All" or "Individual apply" is performed:

- Property group name
- Description
- Display settings

Note: When you change the versions of components that use service components, the properties are updated with the information after the version change.

# 4

## Setting Service Properties

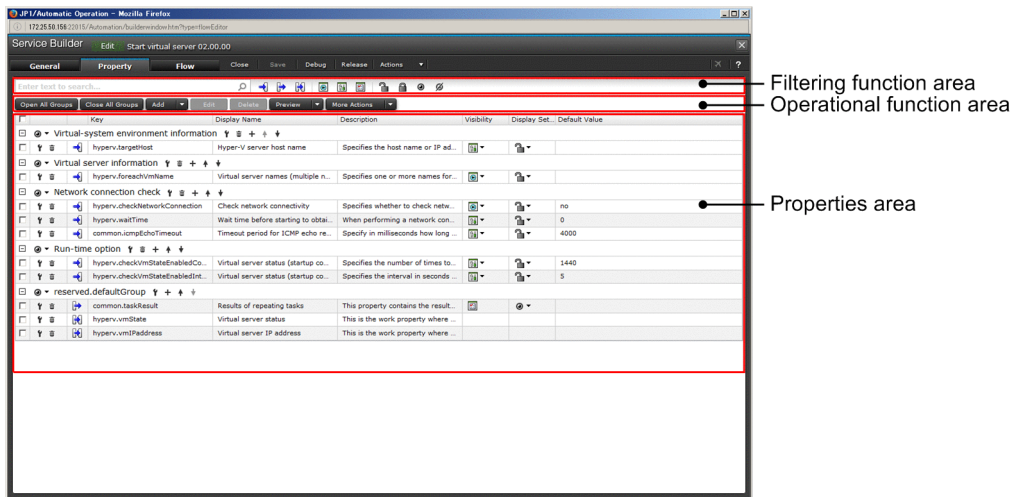
After you edit a flow, set service properties in the **Property** tab of the **Service Builder Edit** window.

After setting service properties, you can define items to be entered when services are set or executed, and items displayed when you check the execution results.

## 4.1 Property tab of the Service Builder Edit window

The **Property** tab of the **Service Builder Edit** window is used to set service properties. To display service properties, perform a creation or edit operation in the **Service Builder Home** window, and then click the **Property** tab in the displayed **Service Builder Edit** window.

Figure 4-1: **Property** tab of the **Service Builder Edit** window



The **Property** tab displays a list of property groups and service properties that belong to each property group. The **Property** tab consists of the following items:

### Filtering function area

The filtering function filters the properties to be displayed when the properties area contains many properties.

### Text search box

You can enter any text string to filter the service properties to be displayed.

### Filtering button

Uses property attributes to filter which service properties are displayed. Clicking this button switches the filtering setting on or off. You can use the following attributes to filter the properties.

- Property types (input property, output property, or variable)
- Visibility of properties (the edit and submit window, edit window only, or the **Task Details** window)
- Display settings of properties (editable, read only, display, or hide)

### Operational function area

You can use the following buttons to perform operations on property groups or service properties:

#### Open All Groups button

Opens all property groups and displays the service properties.

#### Close All Groups button

Closes all property groups.

#### Add button

Enables you to create a property group or service property.

#### Edit button

Enables you to edit the service property selected in the list.

#### Delete button

Deletes the service property selected in the list.

**Preview button**

Displays a preview of the **Create Service** window, **Submit Service** window, or **Task Details** window.

**More Actions button**

Enables you to change the visibility and display settings of the service property selected in the list.

**Properties area**

Displays a list of property groups and service properties. This section describes items displayed in property group lines and service property lines.

The following items are displayed in property group lines:

**Display/Hide pull-down menu**

Displays or hides property groups.

**Edit button**

Enables you to edit a property group.

**Delete button**

Enables you to delete a property group.

**Add button**

Enables you to create a property and then add it to a property group.

**Up Group button and Down Group button**

These buttons change the sort order of property groups.

The following items are displayed in service property lines:

**Check box**

Select this check box to select a service property line. You can select multiple lines.

**Edit button**

Enables you to edit a service property.

**Delete button**

Deletes a service property.

**Service property types**

Displays the service property type (input property, output property, or variable)

**Key**

Displays the service property key.

**Display Name**

Enables you to edit the display name.

**Description**

Enables you to edit the description.

**Visibility pull-down menu**

Enables you to select whether to display the service property in both the **Create Service** window and **Submit Service** window, or only in the **Create Service** window.

**Display Settings pull-down menu**

Enables you to select the display settings for the property.

**Default Value**

Enables you to edit the default value.

## 4.2 Editing and adding service properties

---

Properties defined for a service are called *service properties*. Information set for a service property includes display information, data type, and default value.

### 4.2.1 Overview of service property

A service property is a property used by the user who submits a service to specify a parameter necessary for submitting the service, or to obtain the execution result of the service. The following types of service properties exist:

#### Input property

You can define properties used to store the input values necessary for submitting services. You can define your original properties or use service share properties.

#### Output property

You can define properties used to store the execution results of services.

#### Variable

Variables are temporarily set for inheriting values between plug-ins.

Input properties and output properties are displayed as service properties in the following windows:

- **Create Service** window
- **Submit Service** window
- **Task Details** window
- **Shared Properties Settings** area (for service share properties)



#### Tip

The step properties set to be elevated to service properties in the **Flow** tab of the **Service Builder Edit** window are displayed as service properties in the **Property** tab. You can also create new service properties.

---

### Related topics

- [4.2.2 Procedure for editing service properties](#)
  - [4.2.3 Procedure for adding service properties](#)
  - [3.6.6 Procedure for elevating step properties to service properties](#)
  - [4.3.1 Overview of service share properties](#)
  - [4.3.4 Overview of shared built-in service properties](#)
  - [4.2.7 Visibility and display settings for properties](#)
- 

### 4.2.2 Procedure for editing service properties

Editing a service property is an operation that changes the definition information of the service property. You can change the definition information of service properties in the **Edit Input Property for Service** dialog box, **Edit Output Property for Service** dialog box, or **Edit Variable** dialog box.

## To edit a service property:

1. In the **Properties** tab of the **Service Builder Edit** window, select the service property you want to edit, and then click the **Edit** button.
2. In the displayed dialog box, edit the definition information.

Figure 4-2: **Edit Input Property for Service** dialog box

Specify property attributes.

Key: \* Input

Display Name: \* Input

Description: Enter Description

Property Group: reserved.defaultGroup

Visibility: ☒ Edit and Submit Window ☐ Edit Window Only

Display Settings: ☒ Editable ☐ Read only ☐ Hide

Service Share Property: ☐

Required: ☐

Data Type: string

Default Value: Enter Default Value

Minimum Length: Enter Minimum Length

Maximum Length: Enter Maximum Length

Restricted Character: Enter Restricted Character

\* Required

OK Cancel

3. Click the **OK** button.

## Operation result

The edited information of the service property is set.

## Related topics

- [1.2.3 Procedure for starting editing of service templates](#)
- [4.2.4 Items set for input properties of services](#)
- [4.2.5 Items set for output properties of services](#)
- [4.2.6 Items set for variables](#)

## 4.2.3 Procedure for adding service properties

You can add input properties, output properties, and variables for services. You can map the added service properties to step properties.

## To add a service property:

1. In the **Property** tab of the **Service Builder Edit** window, from the menu of the **Add** button, click **Input Property**, **Output Property**, or **Variable**.



2. In the displayed dialog box, enter the definition information.

Figure 4-3: **Create Input Property for Service** dialog box

**Create Input Property for Service**

Specify property attributes.

Key: \* Enter Key

Display Name: \* Enter Display Name

Description: Enter Description

Property Group: reserved.defaultGroup

Visibility: ☒ Edit and Submit Window ☐ Edit Window Only

Display Settings: ☒ Editable ☐ Read only ☐ Hide

Service Share Property: ☐

Required: ☐

Data Type: string

Default Value: Enter Default Value

Minimum Length: Enter Minimum Length

Maximum Length: Enter Maximum Length

Restricted Character: Enter Restricted Character

\* Required

OK Cancel

3. Click the **OK** button.

**Operation result**

A service property is created and added in the **Property** tab of the **Service Builder Edit** window.



**Tip**

You can also add input properties, output properties, and variables for services from the **Specify Component Input Property Mapping Parameters** dialog box or **Specify Component Output Property Mapping Parameters** dialog box.

**Related topics**

- [4.2.4 Items set for input properties of services](#)
- [4.2.5 Items set for output properties of services](#)
- [4.2.6 Items set for variables](#)

**4.2.4 Items set for input properties of services**

The following table describes the items that can be set in the **Create Input Property for Service** dialog box or **Edit Input Property for Service** dialog box.

Table 4-1: Items set for the input property of a service

Item	Description
Related Step	Displays the related step of the property. This item is displayed when you edit a property for which the <b>GUI Visibility</b> check box is selected in the <b>Step Properties</b> area in the <b>Flow</b> tab of the <b>Service Builder Edit</b> window.
Key	Enter the property key. You cannot change the value for this item when the property has a related step, or when the property is a shared built-in service property.
Display Name <sup>#1</sup>	Enter the property name.
Description <sup>#1</sup>	Enter the description for the property.
Property Group	Select the property group that the property is to belong to. You cannot change the value for this item when the related step of the property is using a service component. You can also set a property group by dragging the service property and dropping it on a property group in the <b>Property</b> tab of the <b>Service Builder Edit</b> window.
Visibility <sup>#1</sup>	Set the visibility in the <b>Create Service</b> window and <b>Submit Service</b> window. Set the visibility when you want to restrict the role of users who are allowed to view the input property. Whether the input property is displayed in each window differs, depending on the combination of the settings for <b>Display Settings</b> and this item. For details about the combination of Visibility and Display Settings, see <a href="#">4.2.7 Visibility and display settings for properties</a> . You cannot change the value for this item when the related step of the property is using a service component, or when the property is a shared built-in service property.
Display Settings <sup>#1</sup>	Select the display settings in the <b>Create Service</b> window and <b>Submit Service</b> window. Set the display settings when you want to restrict the operations the users can perform on the input property. Whether the input property is displayed in each window and whether the value can be changed differ, depending on the combination of the settings for <b>Visibility</b> and this item. For details about the combination of Visibility and Display Settings, see <a href="#">4.2.7 Visibility and display settings for properties</a> .
Service Share Property	If you select this item, the input property becomes a shared service property. You cannot change the value for this item when the property has a related step, or when the property is a shared built-in service property. If <b>composite</b> is selected for the data type, you cannot select this item.
Required	If you select this item, the input property becomes a required property. You cannot change the value for this item when the property has a related step, or when the property is a shared built-in service property.
Data Type	Select the data type of the input property. You cannot change the value for this item when the related step of the property is using a service component, or when the property is a shared built-in service property. In addition, for a basic plug-in, you cannot change the data type of some properties.
Setting Method	Select the setting method of the input property. This item is displayed if all the following conditions are satisfied: <ul style="list-style-type: none"> <li>The property has a related step.</li> <li>The property is not a shared service property.</li> <li>The property is not mapped (in the service component) to another property.</li> </ul> Note that if you select <b>View Property</b> for the setting method, you will not be able to change the default value in the <b>Create Service</b> window or <b>Submit Service</b> window.
Default Value <sup>#1#2</sup>	Specify the default value of the property. This item is not displayed when the property is a shared built-in service property. If the related step of the property being edited is using a service component, and the property is mapped (in the service component) to another property, ( <b>Internal Property</b> ) is displayed as the default value and it cannot be changed.
Validation Script	Specify the validation script of the property. Clicking the <b>Generate</b> button displays an editor used to input the validation script. This item is not displayed for service share properties.

#1

You can also edit the item in the Properties area in the **Property** tab of the **Service Builder Edit** window.

#2

If **composite** is selected for the data type and **Direct Input** is selected for the setting method, you can upload a file for the default value. In this case, even if the property key is specified in the uploaded file, the property key is treated as a character string and the property value cannot be obtained.



### Tip

The difference between the string type and composite type is the number of characters you can specify. For the string type, you can specify 1,024 or fewer characters. For the composite type, you can specify a character string whose size is 30 MB or less. For example, if you want to map an output property that has more than 1,024 characters to the input property of another step, specify the composite type for the data type of the input property. If you select the composite type in the **Create Input Property for Service** dialog box, specify the default value by uploading a file.

The following table lists the items displayed according to the selected data type.

Table 4-2: Items set for the input property of a service (whether the items are displayed differs depending on the data type)

Item	Data type							
	string	boolean	integer	double	date	password	list	composite
Minimum Length	Y	N	N	N	N	Y	N	N
Maximum Length	Y	N	N	N	N	Y	N	N
Minimum Value	N	N	Y	Y	Y	N	N	N
Maximum Value	N	N	Y	Y	Y	N	N	N
Specify List Items	N	N	N	N	N	N	Y	N
Restricted Character <sup>#</sup>	Y	N	N	N	N	Y	N	N

#

Specify the restricted characters in regular expression conforming to PCRE.

Legend:

Y: Displayed. N: Not displayed.

## Related topics

- [4.2.7 Visibility and display settings for properties](#)

## 4.2.5 Items set for output properties of services

The following table lists and describes the items that can be set in the **Create Output Property for Service** dialog box or **Edit Output Property for Service** dialog box.

Table 4-3: Items that can be set for the output property of a service

Item	Description
Related Step	Displays the related step of the property. This item is displayed when you edit a property for which the <b>GUI Visibility</b> check box is selected in the <b>Step Properties</b> area in the <b>Property</b> tab of the <b>Service Builder Edit</b> window.
Key	Enter the property key. You cannot change the value for this item when the related step of the property is using a service component.
Display Name <sup>#</sup>	Enter the property name.
Description <sup>#</sup>	Enter the description for the property.
Property Group	Select the property group that the property is to belong to. You cannot change the value for this item when the related step of the property is using a service component. You can also set a property group by dragging the service property and dropping it on a property group in the <b>Property</b> tab of the <b>Service Builder Edit</b> window.
Display/Hide <sup>#</sup>	Select whether to display the output property in the <b>Task Details</b> window.
Data Type	Select the data type of the output property from string, password, or composite. You cannot change the value for this item when the related step of the property is using a service component.
Default Value <sup>#</sup>	Enter the default value of the output property. The values that can be specified differ depending on the data type. If the property has a related step, this item is displayed as <b>Value</b> , and you can set the property mapping only. If the related step of the property is using a service component, and the property is mapped (in the service component) to another property, (Internal Property) is displayed for <b>Value</b> and it cannot be changed.

#

You can also edit the item in the Properties area in the **Property** tab of the **Service Builder Edit** window.

## 4.2.6 Items set for variables

The following table lists and describes the items that can be set in the **Create Variable** dialog box or **Edit Variable** dialog box.

Table 4-4: Items set for variables

Item	Description
Key	Enter the property key.
Display Name <sup>#</sup>	Enter the property name.
Description <sup>#</sup>	Enter the description for the property.
Data Type	Select the data type of the variable from string, password, or composite.
Default Value <sup>#</sup>	Enter the default value of the variable. The values that can be specified differ depending on the data type.

#

You can also edit the item in the Properties area in the **Property** tab of the **Service Builder Edit** window.

## 4.2.7 Visibility and display settings for properties

You can set whether to display a property in each JP1/AO window or whether the value and display settings can be changed, by using a combination of visibility and display settings of the service property. However, you cannot set these for variables.

### Visibility of properties

You can set visibility for a service property. If you specify visibility for a property, you will be able to display or hide the property in JP1/AO operation windows.

You can display properties in the following JP1/AO operation windows:

- **Create Service** window
- **Submit Service** window
- **Task Details** window
- **Shared Properties Settings** area

For example, you can set to hide properties (that users who submit services do not need to know) from the **Submit Service** window.

Note that the output properties of services are displayed in the **Task Details** window only.

You can set visibility of the input property of a service in the **Property** tab of the **Service Builder Edit** window. You can set **Edit and Submit Window** or **Edit Window Only** for visibility of a property. Visibility is fixed for output properties. You can set (by display settings) whether to display output properties in the **Task Details** window.

#### Edit and Submit Window

Specify this option for the properties you want to display as the input items in the **Create Service** window and **Submit Service** window. If you specify this option for a property, that property is opened to the users with Submit or higher role.

#### Edit Window Only

Specify the option for the properties you want to display as the input items in the **Create Service** window. If you specify this option for a property, that property is opened to the users with Modify or higher role.

### Display settings of properties

If you specify display settings of a service property, you can define whether to display or hide the service property or whether the value can be changed in the **Create Service** window or **Submit Service** window.

#### Editable

If this option is set for the input property of a service, the value for the service property can be changed in the window in which the service property is displayed.

#### Read only

If this option is set for the input property of a service, the value of the service property cannot be changed in the window in which the service property is displayed.

#### Display

If this option is set for the output property of a service, the value of the service property can be viewed in the window in which the service property is displayed.

#### Hide

The service property is not displayed in windows.

The following tables describe whether a property is displayed and operations are enabled in individual windows, for each combination of visibility and display settings.

Table 4-5: Whether properties are displayed in the **Create Service** window

Visibility	Display settings	Whether a property is displayed and operations are enabled in the window
Edit and Submit Window	Editable	The property value and display settings can be changed.
	Read only	The property value and display settings can be changed.
	Hide	The property value and display settings can be changed.
Edit Window Only	Editable	The property value can be changed.
	Read only	The property value is displayed, but cannot be changed.
	Hide	Hidden

Table 4-6: Whether properties are displayed in the **Submit Service** window

Visibility	Display settings	Whether a property is displayed and operations are enabled in the window
Edit and Submit Window	Editable	The property value can be changed.
	Read only	The property value is displayed, but cannot be changed.
	Hide	Hidden
Edit Window Only	Editable	Hidden
	Read only	Hidden
	Hide	Hidden

Table 4-7: Whether properties are displayed in the **Task Details** window

Visibility	Type of the service property (input or output)	Display settings	Whether a property is displayed and operations are enabled in the window
Edit and Submit Window <sup>#</sup>	Input property	Editable	Displayed
		Read only	Displayed
		Hide	Hidden
	Output property	Display	Displayed
		Hide	Hidden
Edit Window Only	Input property	Editable	Hidden
		Read only	Hidden
		Hide	Hidden

#

For an output property, the visibility in the **Property** tab of the **Service Builder Edit** window is displayed as **Output Property**.

Table 4-8: Whether shared service properties are displayed in the **Shared Properties Settings** area

Visibility	Display settings	Whether a property is displayed and operations are enabled in the window
Edit and Submit Window	Editable	The property value can be changed.
	Read only	The property value can be changed.
	Hide	The property value can be changed.
Edit Window Only	Editable	The property value can be changed.
	Read only	The property value can be changed.
	Hide	The property value can be changed.

---

## Related topics

- [4.3 Service share properties](#)
- 

## 4.2.8 Procedure for deleting service properties

You can delete input properties, output properties, and variables of service properties. However, by using this procedure, you can delete only service properties that do not have related steps.

If you want to delete service properties elevated from step properties, in the **Step Properties** area in the **Flow** tab of the **Service Builder Edit** window, clear the **GUI Visibility** check box. You cannot delete service properties contained in service components.

### To delete a service property:

1. In the Properties area in the **Property** tab of the **Service Builder Edit** window, select the service property you want to delete, and click the **Delete** button.

### Operation result

The service property is deleted.

## 4.3 Service share properties

This section describes service share properties, which are a type of service property.

### 4.3.1 Overview of service share properties

Service share properties are properties that are shared by more than one service, each of which can view and update its value. A service share property referenced by a service or task returns a value maintained by the system.

Service share properties that are predefined by JP1/AO are called *shared built-in service properties*.

#### Values set for service share properties

Because the value of a service share property is the same system-wide, you cannot assign different values for different service groups. Even if you create multiple services from a service template and assign each to a different service group, the value of the service share property is shared among the service groups.

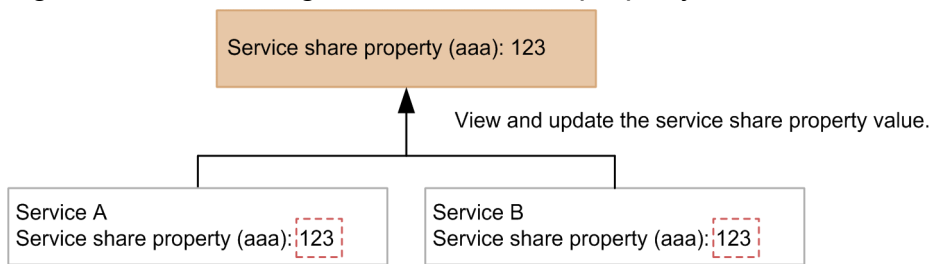
Even if you use multiple service components for which the same service share property is defined in a service template, the service components are displayed as one service share property in the **Create Service** window or **Submit Service** window. For the property name, description, property group, and display settings, the settings for the service share property displayed at the highest level in the **Properties** area in the **Property** tab of the **Service Builder Edit** window apply.

#### Valid range of the values for service share properties

You can also set property values in the **Shared Properties Settings** area of the **Create Service** window, **Submit Service** window, and **Administration** window. However, the property values set in the **Submit Service** window are applied only to the tasks created from the corresponding service. Thus, the settings specified in the **Submit Service** window do not affect the values of the service share properties referenced by other services. Also, even if a property value is changed in the **Shared Properties Settings** area after the service is executed, the property values for the executed service are not affected.

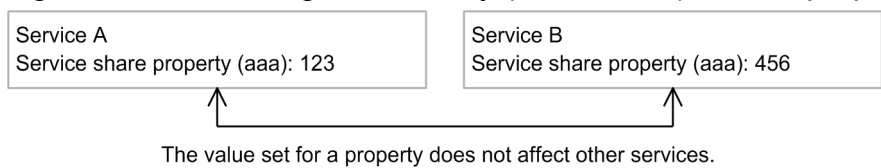
The following figure shows the valid range of a service share property.

Figure 4-4: Valid range of service share property



The following figure shows the valid range of service properties for which no service share property is set.

Figure 4-5: Valid range of ordinary (non-shared) service property





If you do not select the check box in the **Service Share Property** area, the value specified for the property is only valid in the context of that service. You can set property values in the **Create Service** window or **Submit Service** window.

---

### Related topics

- [4.3.4 Overview of shared built-in service properties](#)
  - [Setting Service Share Properties in the JP1/Automatic Operation Administration Guide](#)
- 

## 4.3.2 Procedure for adding service share properties

You can share properties between services by adding a service share property for the properties in a service template. You can also use service share properties predefined in JP1/AO (shared built-in service properties), or add service share properties associated with service templates you have imported.

### To add a new service share property:

1. In the **Property** tab of the **Service Builder Edit** window, click the **Add Input Property** button.
2. In the **Create Input Property for Service** dialog box, in the **Service Share Property** area, select the check box.

### To set a defined service property as a service share property:

1. In the **Property** tab of the **Service Builder Edit** window, click the **Edit** button for the input property of a service.
2. In the **Edit Input Property for Service** dialog box, in the **Service Share Property** area, select the check box.

However, you cannot change the selection of the **Shared Properties Settings** check box in the following cases:

- The service has a related step.
- The property is a shared built-in service property.
- **composite** is selected for the data type.

### To add a defined service share property:

1. In the **Property** tab of the **Service Builder Edit** window, select **Add > Service Share Property**.
  2. In the **Select Service Share Property** dialog box, select the service share property you want to add and the property group to which the service share property is to be added, and then click the **OK** button.
- 

### Related topics

- [4.3.1 Overview of service share properties](#)
  - [4.3.3 Notes on defining service share properties](#)
  - [4.3.4 Overview of shared built-in service properties](#)
- 

## 4.3.3 Notes on defining service share properties

Note the following when defining service share properties:

## Setting values for service share properties

- The name and description you specify for properties in the **Property** tab of the **Service Builder Edit** window appear in the **Create Service** window and **Submit Service** window.
- Create a property key that is unique within the system by specifying the domain name in reverse order from the top level as a period-separated value. If you choose not to use domain names for property keys, make sure that the property key you specify is not being used for another property whose value you do not want shared.
- If you need to change a parameter of a service share property other than the property name or description, use the following procedure:
  1. Delete from JP1/AO all service templates that include the service share property you want to modify.
  2. Add the service templates to JP1/AO with the new values specified for the service share property.The value of the service share property is now changed.
- The following describes what happens when a service share property with the same key as an existing service share property is assigned to a service template:
  - Initially, the service share property is assigned the value specified in the **Default Value** for the service share property defined in the service template that was built, released, or imported. You can then specify a value for the property in the **Shared Properties Settings** area, **Create Service** window, or **Submit Service** window.
  - The property name, description, and default value can differ from that of the existing service share property. However, if an item other than the property name, description, or default value differs, an error occurs when you attempt to build, release, or import the service template.
  - When a value differs from that of the existing service share property, the specified value only appears in windows and dialog boxes that display the property name or description as a property of the service template.
- Service share properties selected in the **Select Service Share Property** dialog box are added with no default value specified. Set the default value in the **Edit Input Property for Service** dialog box.

## Assigning a service share property to a property group

- You can assign service share properties to any property group when defining a service template. The same service share property can belong to a different property group in different service templates.

### 4.3.4 Overview of shared built-in service properties

Shared built-in service properties are properties that are predefined in the JP1/AO system. Unlike other properties, shared built-in service properties are not tied to a specific service or task. JP1/AO functions that reference shared built-in service properties use the value assigned to the property when they execute.

You can also define a shared built-in service property as a service share property of a service, and reference it from within a task. In this case, the values while services are executed are referenced in the same manner as other service share properties.

You can define a shared built-in service property as a service share property of a service in the **Property** tab of the **Service Builder Edit** window. If you select **Add > Service Share Property** in the **Property** tab of the **Service Builder Edit** window, a list of shared built-in service properties appears.

The following table lists the shared built-in service properties provided in JP1/AO.

Table 4-9: List of shared built-in service properties

No.	Property key	Property name <sup>#</sup>	Description <sup>#</sup>
1	com.hitachi.software.dna.sys.mail.notify	Email notification	Enables or disables the email notification functionality. (Built-in shared service property)
2	com.hitachi.software.dna.sys.mail.smtp.server	SMTP server address	Specifies the SMTP server address. The address can be specified as an IPv4 or IPv6 address, or as a host name. Only one of the above can be specified. Multiple addresses cannot be specified by separating them with commas. (Built-in shared service property)
3	com.hitachi.software.dna.sys.mail.smtp.port	SMTP server port number	Specifies the SMTP server port number. (Built-in shared service property)
4	com.hitachi.software.dna.sys.mail.smtp.userid	SMTP server user ID	Specifies the user ID of the user who logs in to the SMTP server. (Built-in shared service property)
5	com.hitachi.software.dna.sys.mail.smtp.password	SMTP server password	Specifies the password of the user who logs in to the SMTP server. (Built-in shared service property)
6	com.hitachi.software.dna.sys.mail.from	Notification email sender	Specifies the sender of notification emails. (Built-in shared service property)
7	com.hitachi.software.dna.sys.mail.to	Notification email recipients (To)	Specifies the "To" recipients of notification emails. Multiple email addresses can be specified by separating them with commas. (Built-in shared service property)
8	com.hitachi.software.dna.sys.mail.cc	Notification email recipients (Cc)	Specifies the "Cc" recipients of notification emails. Multiple email addresses can be specified by separating them with commas. (Built-in shared service property)
9	com.hitachi.software.dna.sys.mail.bcc	Notification email recipients (Bcc)	Specifies the "Bcc" recipients of notification emails. Multiple email addresses can be specified by separating them with commas. (Built-in shared service property)
10	com.hitachi.software.dna.sys.task.log.level	Task log output level	Specifies the level of messages output to the task log. (Built-in shared service property)
11	com.hitachi.software.dna.sys.ssh.privatekey.passphrase	Pass phrase of the private key (for SSH public key authentication)	Specifies the pass phrase of the private key used for SSH public key authentication. (Built-in shared service property)

#

You can change the contents of the **Create Service** dialog box and the **Submit Service** dialog box by entering a property name and description of your choice for the built-in shared service property. However, the information displayed in the **Shared Properties Settings** area is not changed from its initial state at installation.

The following table describes detailed information about each property.

Table 4-10: Detailed information about shared built-in service properties

No.	Property key	Data type	Default value	Required?	Length		Specifiable characters	Values in list
					Minimum	Maximum		
1	com.hitachi.software.dna.sys.mail.notify	boolean	false	true	--	--	--	--

No.	Property key	Data type	Default value	Required?	Length		Specifiable characters	Values in list
					Minimum	Maximum		
2	com.hitachi.software.dna.sys.mail.smtp.server	string	--	false	0	255	No restrictions	--
3	com.hitachi.software.dna.sys.mail.smtp.port	integer	25	false	--	--	--	--
4	com.hitachi.software.dna.sys.mail.smtp.userid	string	--	false	0	255	No restrictions	--
5	com.hitachi.software.dna.sys.mail.smtp.password	password	--	false	0	1,024	No restrictions	--
6	com.hitachi.software.dna.sys.mail.from	string	--	false	0	255	No restrictions	--
7	com.hitachi.software.dna.sys.mail.to	string	--	false	0	255	No restrictions	--
8	com.hitachi.software.dna.sys.mail.cc	string	--	false	0	255	No restrictions	--
9	com.hitachi.software.dna.sys.mail.bcc	string	--	false	0	255	No restrictions	--
10	com.hitachi.software.dna.sys.task.log.level	list	10	true	--	--	--	0,10,20,30,40
11	com.hitachi.software.dna.sys.ssh.privatekey.passphrase	password	--	false	0	1,024	No restrictions	--

Legend:

--: Not applicable.

## 4.4 Setting property groups

You can add and edit property groups that service properties belong to.

### 4.4.1 Procedure for setting property groups

In the **Property** tab of the **Service Builder Edit** window, add and edit property groups.


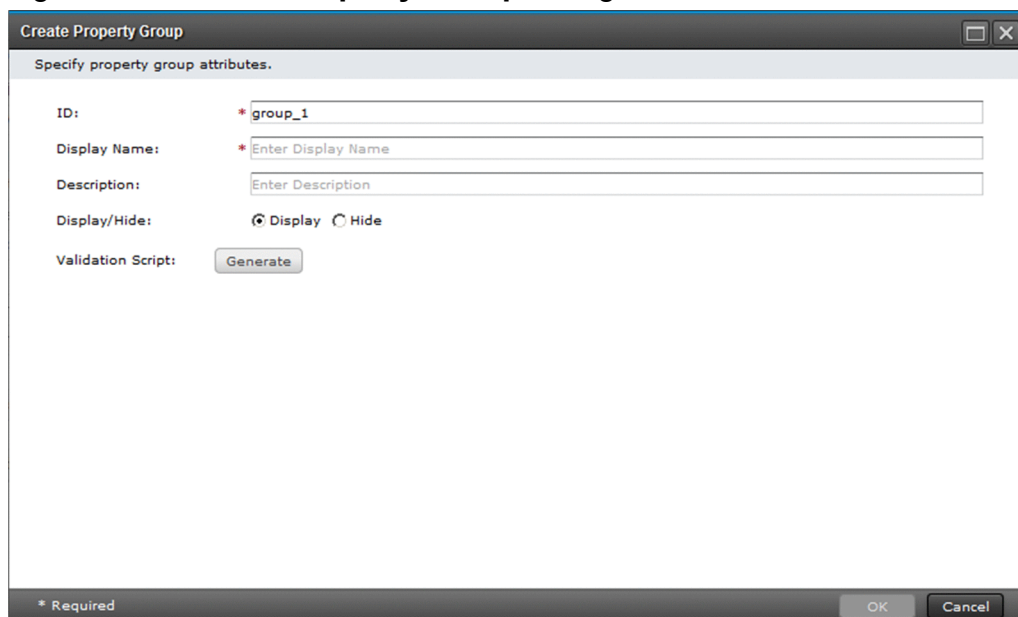
1. In the **Property** tab of the **Service Builder Edit** window, select **Add > Property Group** to add a property group, or click the **Edit** button  for the line of a property group to edit the property group.
2. In the dialog box that appears, set the definition information of the property group.

Figure 4-6: **Create Property Group** dialog box

The image shows a 'Create Property Group' dialog box. It has a title bar with 'Create Property Group' and standard window controls. The main area is titled 'Specify property group attributes.' and contains several fields: 'ID:' with a text box containing 'group\_1', 'Display Name:' with a text box containing 'Enter Display Name', 'Description:' with a text box containing 'Enter Description', 'Display/Hide:' with radio buttons for 'Display' (selected) and 'Hide', and 'Validation Script:' with a 'Generate' button. A legend at the bottom left indicates that an asterisk (\*) denotes a required field. At the bottom right are 'OK' and 'Cancel' buttons.

The following table lists and describes the items that can be set.

Table 4-11: Items set for property groups

Item	Description
Related Step	Displays the related step of the property group.
ID	Enter the ID of the property group. You cannot change the value for this item when the related step of the property group is using a service component.
Display Name	Enter the name of the property group.
Description	Enter the description for the property group.
Display/Hide	Select whether to display the property group in the following windows: <ul style="list-style-type: none"><li>• <b>Create Service</b> window</li><li>• <b>Submit Service</b> window</li><li>• <b>Task Details</b> window</li></ul>
Custom File package	Specify a custom file package. This item does not appear if a related step exists when editing a property group.
Validation Script	Specify the validation script for the properties.

Item	Description
Validation Script	Click the <b>Generate</b> button to display the editor for entering the validation script. This item does not appear if a related step exists when editing a property group.

3. Click the **OK** button.

## Operation result

The property group is set.

### 4.4.2 Procedure for deleting property groups

You can delete property groups that service properties belong to. The service properties that belonged to the deleted property groups will belong to "default property".



#### Tip

You cannot delete the "default property" property group and property groups in related steps.

#### To delete a property group:

1. In the **Property** tab of the **Service Builder Edit** window, click the **Delete** button for the line of the property group you want to delete.

## Operation result

The property group is deleted.

# 5

## Managing Service Templates

This chapter describes operations (other than the edit operation) that can be performed on service templates. For example, you can copy or release service templates.

## 5.1 Viewing service templates

---

You can view the settings of development service templates and release service templates.

### 5.1.1 Procedure for viewing service templates

You can view the settings of a development service template or release service template in another window while you edit the flow in a service template. Because the reference window is opened as read-only, there is no risk of the user inadvertently changing the existing templates.

You can also view the settings of a service template in the **Service Builder Home** window and **Service Template** window. In the **Service Template** window, you can view release service templates only.

#### To open a service template for reference purposes:

1. In the **Service Builder Home** window, select the service template you want to view on the **Developing** tab or **Released** tab, and then click the **View** button.

#### Operation result

The **Service Builder View** window appears.



## 5.2 Copying service templates

You can copy a development service template or release service template.

### 5.2.1 Procedure for copying service templates

You can copy a development service template or release service template and create a new development service template that retains the settings of the original. You can use this procedure when developing a new service template based on an existing service template, or when creating an upgraded version of an existing service template.

#### ! Important

If you copy a service template that contains a step using a service component, make sure that the release service template that the service component is based on has been imported to the JP1/AO server. If such a release service template does not exist on the JP1/AO server, you will not be able to obtain information about the service component even after the service template is copied.

#### To copy a service template:

1. In the **Service Builder Home** window, select the service template you want to copy on the **Developing** tab or **Released** tab, and then click the **Copy** button.
2. In the **Copy Service Template** dialog box, set the definition information for the service template, and then click the **OK** button.

Figure 5-1: **Copy Service Template** dialog box

Copy Service Template

The vendor ID and vendor name were deleted because the service template on the copy source is using a reserved vendor ID.

Specify service template attributes. You should change at least one of the following:

1. Service Template Key Name (or)
2. Service Template Version (or)
3. Vendor ID

Key Name: \* getInfoVMhyperV

Version: \* 02.00.00

Vendor ID: \* Enter Vendor ID

Display Name: \* Obtain the virtual server information list

Vendor Name: Enter Vendor Name

Description: Obtains the virtual server information list in the Hyper-V environment.

Tags: Gather VM information x Hyper-V 2008 x

\* Required

OK Cancel

#### Operation result

The service template is copied, and the **Flow** tab of the **Service Builder Edit** window appears. You can continue to create and edit flows, and set service properties.

You cannot copy a service template without changing at least one of the vendor ID, service template ID, and service template version. These three items together ensure that the service template can be uniquely identified within the JP1/AO system. You cannot specify a vendor ID that begins with com.hitachi.software.dna. This string is reserved in

JP1/AO. If the vendor ID of the service template you are copying begins with com.hitachi.software.dna, the vendor ID and vendor name are deleted when you copy the template.



### Tip

You can also copy release service templates in the **Service Templates** window by clicking the **Copy** button.

---

## Related topics

- [2.2.4 Items to set in service template definition information](#)
  - [5.2.2 Uniqueness of service templates and plug-ins](#)
- 

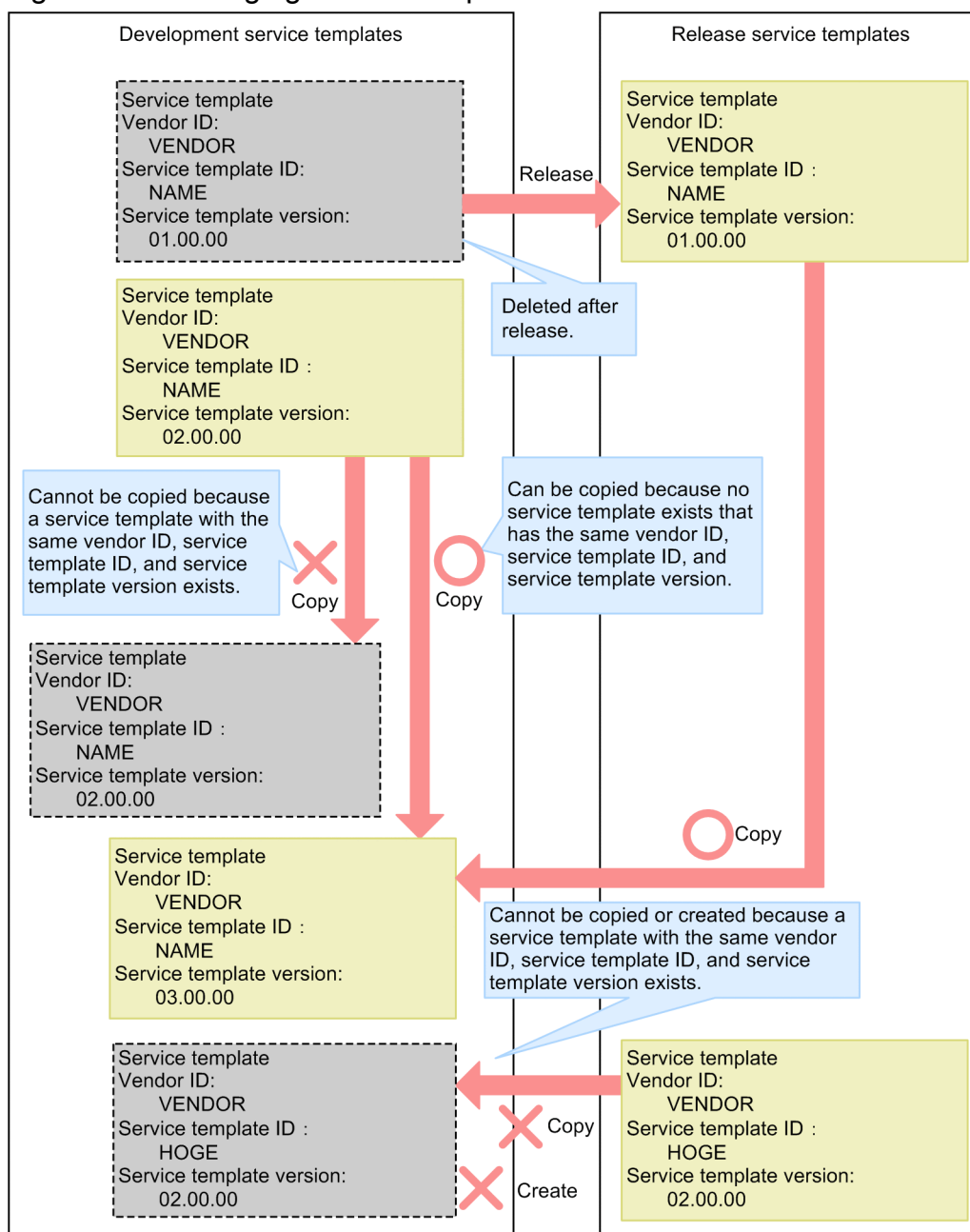
## 5.2.2 Uniqueness of service templates and plug-ins

The following three items ensure that service templates and plug-ins can be uniquely identified within the JP1/AO system:

- Vendor ID
- Service template ID (or plug-in ID)
- Service template version (or plug-in version)

These three items are centrally managed with no distinction made between development service templates (development plug-ins) and release service templates (release plug-ins). For this reason, you cannot create or copy a service template or plug-in whose vendor ID, service template ID (plug-in ID), and service template version (plug-in version) all match those of an existing release service template (or release plug-in).

Figure 5-2: Managing service template versions



## 5.3 Deleting development service templates

---

You can delete development service templates.

### 5.3.1 Procedure for deleting development service templates

When you delete a development service template, the definition of the service template is deleted from the JP1/AO server and from the list under **Service Template List**.

#### Important

You can delete a development service template that another user is editing. Make sure that the service template is not being edited by another user before you delete it.

#### To delete a development service template:

1. On the **Developing** tab of the **Service Builder Home** window, select the service template you want to delete.
2. From the **More Actions** pull-down menu, select the **Delete** button.

#### Operation result

The development service template is deleted.

#### Tip

- The system does not delete the development plug-ins and release plug-ins used by the development service template you are deleting, regardless of whether they are used by another service template. If a development plug-in or release plug-in is no longer needed, you can delete it individually.
- JP1/AO deletes any services that were created by building the development service template you are deleting, and archives any tasks generated from those services. If a task is in progress, the deletion processing fails. If any debug services or debug tasks created while debugging the development service template remain in the system, those services and tasks are deleted. If a debug task is still in progress, the deletion processing fails.
- If the development service template you are deleting has already been released by another user, the deletion processing fails.
- To delete release service templates, perform operations in the **Service Templates** window. For details about the procedure, see *Deleting service templates* in the *JP1/Automatic Operation Administration Guide*.

---

#### Related topics

- [7.2.1 Procedure for deleting plug-ins](#)
  - [1.2.4 Notes when an interrupt operation is performed in the Service Builder window](#)
  - [Deleting service templates in the JP1/Automatic Operation Administration Guide](#)
-

## 5.4 Releasing service templates

---

After developing service templates and validating them, you can release the service templates.

### 5.4.1 Overview of service template release

Release is the process of making a service template available to other users after it has undergone validation processing. When successfully released, a service template is packaged and can be created as a service.

Note that you cannot edit a service template or its plug-ins after the service template has been released. To edit a released service template or its plug-ins, you need to copy the service template or plug-in and edit the copy.

#### Objective

Perform a release operation when you want to use a service template in the active environment. A released service template is packaged as a service template of the Release configuration type, and imported to the JP1/AO server.

#### Number of releases

You can release a service template only once. When you release a service template, the pre-release development service template is deleted. Any services that were created from the development service template prior to release are deleted, and associated tasks are archived. Debug services and debug tasks are deleted.

For details about the specific service templates, services, tasks, debug services, and debug tasks that are deleted and archived during a release operation, see [A.1\(5\) Deletion and archiving of service templates, services, and tasks during build and release operations](#).

#### Assigned configuration type

After its release, a service template is assigned the Release configuration type. In addition to users in the Admin and Develop roles, users in the Modify and Submit roles can configure and execute release service templates and the associated services and tasks that were created. Note that you cannot edit a service template or its plug-ins after the service template has been released. To edit a released service template or its plug-ins, you need to copy the service template or plug-in and edit the copy. For details about the configuration types assigned to service templates and plug-ins by the release processing, see [A.1\(2\) Configuration types assigned to service templates and plug-ins by build and release operations](#).

#### Output destinations of service templates

When you release a service template, a service template package is created in the folder shown below with the name *vendor-ID\_name\_version.st*.

You can also save a service template package in any folder by exporting the released service template.

In a Windows non-cluster system

*JP1/AO-installation-folder\develop\output*

In a Linux non-cluster system

*/var/opt/jp1ao/develop/output*

In a Windows cluster system

*shared-folder-name\develop\output*

In a Linux cluster system

shared-folder-name/develop/output

---

## Related topics

- [5.4.2 Procedure for releasing a service template](#)
  - [2.1 Overview of development service templates and release service templates](#)
  - [5.2.1 Procedure for copying service templates](#)
  - [\(1\) Structure of build and release processes and how they differ](#)
  - [\(2\) Configuration types assigned to service templates and plug-ins by build and release operations](#)
  - [\(5\) Deletion and archiving of service templates, services, and tasks during build and release operations](#)
- 

## 5.4.2 Procedure for releasing a service template

When you select and release a service template, a package is created based on the service template and imported to the JP1/AO server. If the development environment is the same as the active environment, you are then able to create the service.

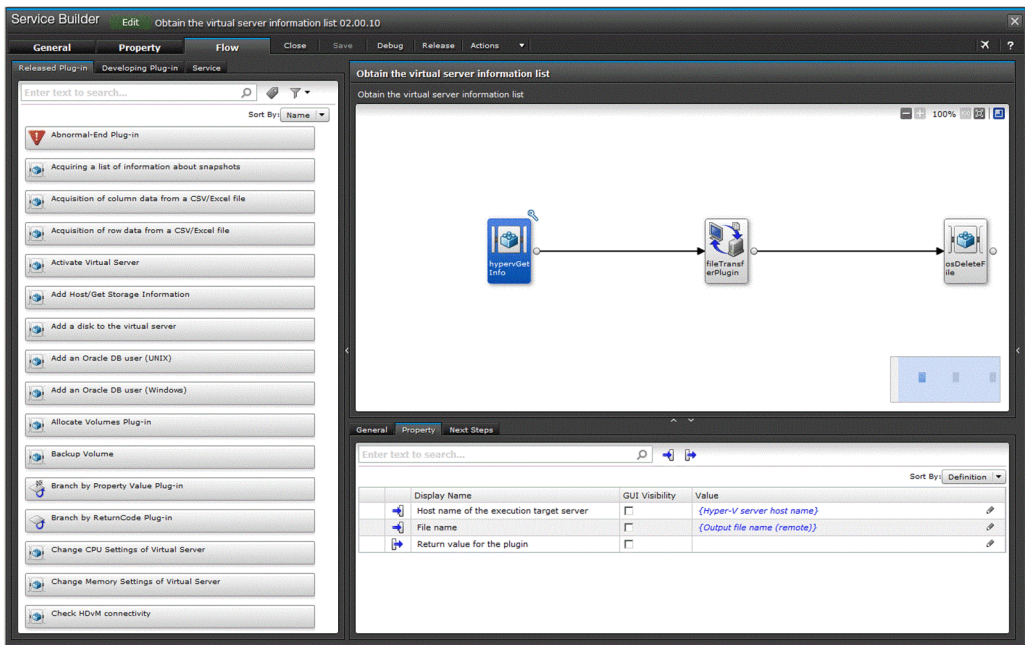
### Important

- If a service template has been released, it cannot be re-edited. If you want to edit a released service template, copy it so that it can be edited.
- If a service template contains a step that uses a service component, make sure that the release service template that the service component is based on has been imported to the JP1/AO server. If the release service template has been deleted or has not been imported, the release processing fails.

### To release a service template:

1. On the **Developing** tab of the **Service Builder Home** window, select the service template you want to release, and then click the **Edit** button.
2. In the **Service Builder Edit** window, click the **Release** button.

Figure 5-3: **Service Builder Edit** window



- 3. In the confirmation window, click the **OK** button.
- 4. In the **Build / Release Result** dialog box, check the result, and then click the **Close** button.

Operation result

The service template is released. The release service template is displayed in the **Service Templates** window.

If the service template has been deleted or released by another user, an error occurs and the release processing fails.

If an error occurs during the release processing, a message is displayed to the operator that describes the cause of the error and instructs the operator to fix the flow. Then, at the upper right of the **Service Builder Edit** window, the **Error** button is displayed. After you close the message display, you can view the message again by clicking the **Error** button. This button is retained in the **Service Builder Edit** window until the operator closes this window.

Related topics

- [5.4.1 Overview of service template release](#)
- [\(5\) Deletion and archiving of service templates, services, and tasks during build and release operations](#)



## 5.5 Exporting service templates

---

When you want to import to the active environment a service template created in the development environment, first export the service template package to any folder you want.

### 5.5.1 Procedure for exporting service templates

After you export a service template that has been built or released, you can save the service template package in any folder. The following describes the procedure for exporting a developing service template.

#### To export a developing service template:

1. On the **Developing** tab of the **Service Builder Home** window, select the service template you want to export.
2. From the **More Actions** pull-down menu, select **Export**.
3. Specify any folder you want, and then click the **Save** button.

#### Operation result

The service template package is saved in a folder.



#### Tip

You can also export a release service template by selecting **Export** from the **More Actions** pull-down menu in the **Service Templates** window. The service template package is saved in the folder you select in your Web browser.

---

#### Related topics

- [5.6 Importing service templates](#)
-



## 5.6 Importing service templates

---

When you want to move a service template you developed to another environment, import the service template file to the destination environment.

### 5.6.1 Procedure for importing service templates

When you want to move a service template you developed to another environment, you need to import the exported service template file to the destination environment.

#### Important

If you import a service template from the **Service Builder Home** window or the **Service Templates** window, the file size must be 100 MB or less.

#### Tip

We recommend that you use different environments for the development environment for service templates and the active environment. If the development environment and the active environment are different, use the service templates created in the development environment after importing them to the active environment.

When the development environment and the active environment are different, we recommend that you check the behavior of a service template in the development environment before you actually use the service template in the active environment. After checking the behavior in the development environment, copy the service template package to the active environment, and then import the service template.

#### To import a service template:

1. On the **Developing** tab of the **Service Builder Home** window, click the **Import** button.
2. In the **Import Service Template** dialog box, click the **Browse** button.
3. Select a service template file, and then click the **Import** button.

#### Operation result

A service template is imported to JP1/AO.

---

#### Related topics

- [5.6.2 Importing service templates that contain steps using service components](#)
  - [5.5 Exporting service templates](#)
  - [5.6.3 Reason for maintaining separate development and active environments](#)
-

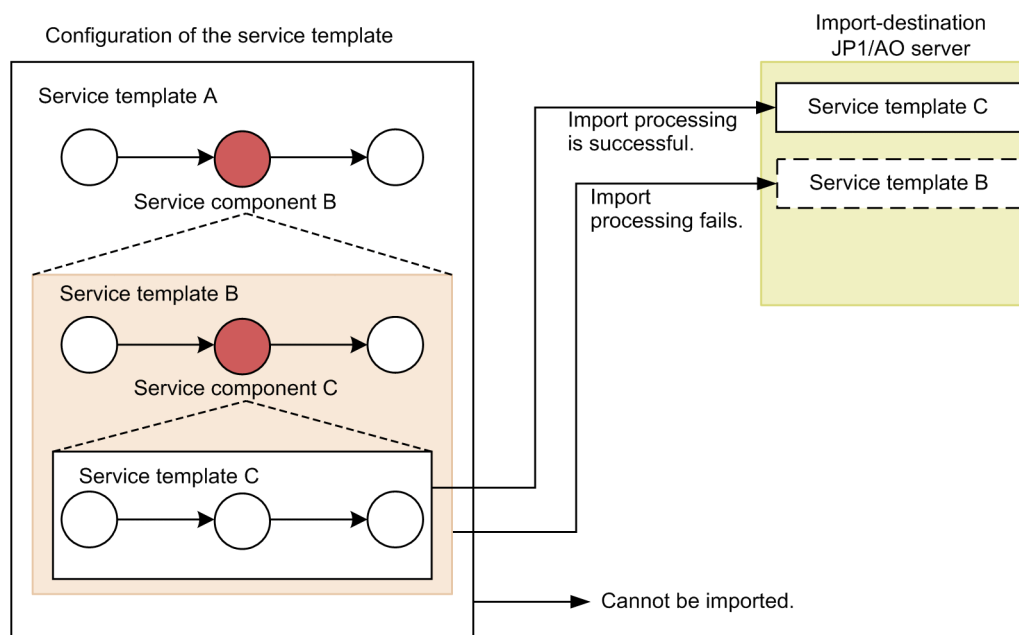
## 5.6.2 Importing service templates that contain steps using service components

If a service template you import contains a step using a service component, the release service template that the service component is based on is also imported.

Service templates are imported from deeper hierarchy levels. If importing one of the service templates fails, an import error occurs, and the import processing stops. However, the service templates that have already been imported when the processing stops remain normally imported.

The following figure shows the behavior when importing a release service template (that a service component in the service template being imported is based on) fails.

Figure 5-4: Behavior when importing a release service template that a service component is based on fails



In this figure, importing Service template B (that Service component B is based on) fails. In this situation, Service template C (which was imported before Service template B) has already been imported. Service template B (for which import processing failed) and Service template A (which was planned to be imported after Service template B) cannot be imported.

## 5.6.3 Reason for maintaining separate development and active environments

When you develop a service template in a separate environment from the active environment, each environment has its own set of service share properties.

When the development and active environments are the same

A given service share property has one value within the JP1/AO system. When you change the value of a service share property, the change affects release service templates in addition to development service templates.

### When the development and active environments are different

The values of service share properties are managed separately in the active and development environments. When you change the value of a service share property, the change only applies to the service share property associated with the development service template. It does not affect the service share property associated with the release service template.

Therefore, we recommend that you keep the development and active environments of the service template separate. This allows you to prevent changes made during development from affecting the service share properties of the release service template.

# 6

## Creating and editing plug-ins

This chapter describes how to create and edit plug-ins. You can use the plug-ins provided by JP1/AO in an unmodified state, or create and edit plug-ins to define processing that meets a specific need.

## 6.1 Overview of plug-ins

A plug-in defines processing that executes a task.

There are three types of plug-ins in JP1/AO: basic plug-ins, release plug-ins, and development plug-ins. In the **Component** area of the **Flow** tab of the **Service Builder Edit** window, basic plug-ins and release plug-ins are displayed in the **Release** tab, and development plug-ins are displayed in the **Develops** tab.

For the sake of expedience, plug-ins are separated into basic plug-ins and content plug-ins according to their origin. For details, see *Types of Service Templates and Plug-ins* in the *JP1/Automatic Operation Service Template Reference*.



### Tip

In the **Component** area of the **Flow** tab of the **Service Builder Edit** window, the **Services** tab displays service components. Service components are release service templates that can be placed as steps in the **Flow** area. Thus, service components are not included in the plug-ins described in this manual.

Table 6-1: Types of plug-in

Type		Description
Basic plug-in		<ul style="list-style-type: none"><li>Displayed in the <b>Release</b> tab.</li><li>A plug-in provided by JP1/AO. A basic plug-in defines generic processing like email notification and flow repetition.</li></ul>
Content plug-in	Release plug-in	<ul style="list-style-type: none"><li>Displayed in the <b>Release</b> tab.</li><li>A plug-in that is imported into JP1/AO by a user releasing a service the user created.</li><li>A plug-in in a service template provided by JP1/AO.</li><li>A plug-in that is imported into the JP1/AO server in released configuration, is also handled as a release plug-in.</li></ul>
	Development plug-in	<ul style="list-style-type: none"><li>Displayed in the <b>Develops</b> tab.</li><li>A plug-in that a user created as a new plug-in, which has not yet been released. A plug-in that is being created based on a copy of an existing plug-in is also classified as a development plug-in.</li><li>When you build a development service template that includes a development plug-in, the development plug-in is imported into the JP1/AO server and can be executed for testing purposes.</li><li>A plug-in that is imported into the JP1/AO server in debug configuration, is also handled as a development plug-in.</li></ul>

By using plug-ins, you can perform actions like the following:

- Send notification emails and control flow repetition.
- Transfer files and folders between the JP1/AO server and a remote host.
- Connect to a remote host and execute commands and scripts.

In JP1/AO, a user can create a custom content plug-in. Users can also create plug-ins that connect to a remote host and execute commands and scripts, and incorporate these plug-ins into a service template.

When JP1/AO executes a content plug-in, it uses WMI to connect to operation target devices that are running Windows, and SSH to connect to UNIX devices. For details about basic plug-ins, see the description of basic plug-ins in the *JP1/Automatic Operation Service Template Reference*.

---

## Related topics

- [6.1.1 Available operations by plug-in type](#)
  - [6.1.2 Plug-in executors](#)
  - [6.1.3 Files transferred to Windows systems](#)
  - [6.1.4 Files transferred to UNIX systems](#)
  - [6.1.5 Commands required for plug-in execution](#)
  - [6.1.6 Locale set for operation target devices during plug-in execution](#)
  - [6.1.7 Character set used for communication by JP1/AO during plug-in execution](#)
  - [6.1.8 Setting a specific character set during plug-in execution](#)
  - [\(2\) Configuration types assigned to service templates and plug-ins by build and release operations](#)
- 

### 6.1.1 Available operations by plug-in type

The operations you can perform depend on the type of plug-in selected.

Table 6-2: Available operations by plug-in type

Type	Plug-in operation				
	Show in list	Create	Edit	Delete	Copy
Basic plug-in	Y	N	N	N	N
Development plug-in	Y	Y	Y	Y #1	Y
Release plug-in	Y	N #2	N	Y #1, #3	Y

Legend:

Y: Can be performed. N: Cannot be performed.

#1

The plug-in cannot be deleted when being used in a development service template.

The plug-in cannot be deleted if a development service template has been built that incorporates the plug-in you want to delete as a step. In this case, in the relevant development service template, delete the step that uses the plug-in and build the template again. You can then delete the plug-in.

#2

If a development service template that incorporates a development plug-in is released, the development plug-in becomes a release plug-in.

#3

The plug-in cannot be deleted when being used in a release service template.

### 6.1.2 Plug-in executors

The executor of a plug-in depends on the combination of the setting of the local execution function and the operation target device.

## When local execution function is disabled

When local execution function is disabled, the executor of a plug-in is as follows:

- When the OS of the operation target device is Windows

In most circumstances, the execution user of a command or script is the user who connects to the operation target device. However, you can also execute commands or scripts with the permissions of the System account after connecting to the operation target device.

When the operation target device is running Windows, user profiles are not inherited. This means a plug-in can produce different execution results from a command or script executed on the desktop.

To avoid this issue, do not reference settings in user profiles, such as user environment variables, registry entries, and Internet Explorer settings, when executing 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 execute a command or script that references Internet Explorer proxy settings, the command or script might fail with a communication error. This might occur in scenarios such as when implementing a Windows Update using a script.

- When the OS of the operation target device is UNIX

Generally, the user who connects to the operation target device is the executor of commands and scripts. JP1/AO also provides a function that allows you to elevate the executor of a command or script to root privileges.

Note that when a user connects to an operation target device as a user with root privileges, the connection of the root privileged user must be permitted on the operation target device side.

The following table lists the executors of plug-ins.

Table 6-3: Execution users for plug-ins

Plug-in	Elevation to root privileges <sup>#1</sup>	User who connected to operation target device	Executor of command or script <sup>#2</sup>
<ul style="list-style-type: none"><li>• Basic plug-in (general command plug-in, file-transfer plug-in, or terminal connect plug-in)</li><li>• Content plug-in</li></ul>	Enabled	User with root privileges	User with root privileges
		User without root privileges	User with root privileges
	Not enabled	User with root privileges	User with root privileges
		User without root privileges	User without root privileges

<sup>#1</sup> The process by which the user is elevated to root privileges depends on the plug-in.

- For general command plug-in and file-transfer plug-in:

You can specify whether to elevate the user to root privileges in the plug-in properties. For details about the elevation of users to root privileges, see the section describing basic plug-ins in the manual *JP1/Automatic Operation Service Template Reference*.

- For the terminal connect plug-in:

You cannot configure JP1/AO to elevate users of the terminal connect plug-in to root privileges. To achieve this, you need to execute the command that elevates the user to root privileges in a terminal command plug-in. For details about the elevation of users to root privileges, see the section describing basic plug-ins in the manual *JP1/Automatic Operation Service Template Reference*.

- For content plug-ins:

You can specify the permissions of the executor by using the **Execute with root privileges** check box on the **Remote Command** tab of the **Create Custom Plug-in** or the **Edit Custom Plug-in** dialog box. If you select this check box, commands and scripts are executed as a user with root privileges. If the check box is not selected, commands and scripts are executed with the permissions of the user who connected to the operation target device.



## Tip

When the **Execute with root privileges** check box is selected, how you specify the superuser password depends on the credential type specified on the **remote Command** tab in the **Create Custom Plug-in** or **Edit Custom Plug-in** dialog box.

- If **Shared agentless setting** is selected as the credential type for the plug-in  
JP1/AO uses the superuser password specified in the definition of the connection destination.
- If **Service input property** is selected as the credential type for the plug-in  
JP1/AO uses the superuser password specified in the plugin.suPassword plug-in property.

#2 In the case of a file-transfer plug-in, this executor is the user who transfers the file. Also, in the case of a terminal connect plug-in, the command is actually executed by a terminal command plug-in.

## When local execution function is enabled

When the local execution function is enabled and the operation target device is the local host, the executors of plug-ins are as follows :

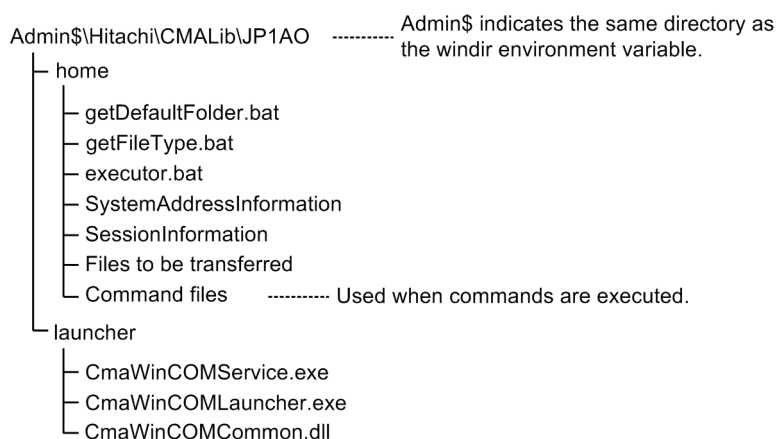
- When the OS of the local host is Windows  
Commands and scripts are executed by a user with System account privileges.
- When the OS of the local host is UNIX  
Commands and scripts are executed by a user who has root-user privileges.

Note that the executor for the operation target device that is other than the local host is the same executor as when the local execution function is disabled.

## 6.1.3 Files transferred to Windows systems

When a general command plug-in, file-transfer plug-in, or content plug-in executes an operation on a Windows device, JP1/AO transfers the files shown in the following figure to the device. The files are deleted when the plug-in finishes executing.

Figure 6-1: Files transferred to Windows systems



Files are transferred when either of the following conditions are met:



- You use a file-transfer plug-in.
- A script is executed in a content plug-in.

However, when the local execution function is enabled and the operation target is the local host, the file is not transferred or copied to the local host.

## 6.1.4 Files transferred to UNIX systems

When a file-transfer plug-in or content plug-in executes an operation on a UNIX device, JP1/AO transfers the files as listed in the following figure to the device. The files are deleted when the plug-in finishes executing.

Figure 6-2: Files transferred to UNIX systems

### **Working folder**

└ Files to be transferred

Files are transferred in the following circumstances:

- You use a file-transfer plug-in.
- A script is executed in a content plug-in.

However, when the local execution function is enabled and the operation target is the local host, the file is not transferred or copied to the local host.

You can set *working-folder* by using the `plugin.remoteCommand.workDirectory.ssh` key in the user-specified properties file (`config_user.properties`). The default is `/tmp/Hitachi_AO`.

---

### Related topics

- User-specified properties file (`config_user.properties`) in the JP1/Automatic Operation Configuration Guide
- 

## 6.1.5 Commands required for plug-in execution

Certain commands must be installed in the operating system of the operation target device before you can execute plug-ins. For details, see the Release Notes.

## 6.1.6 Locale set for operation target devices during plug-in execution

The locale setting that applies to a device on which an operation is performed by a plug-in depends on the operating system. Below are descriptions of the locale settings applied when plug-ins are executed in each operating system.

### In Windows

When JP1/AO executes a script or command on an operation target device, make sure that the locale and character set of the operation target device match those of the JP1/AO server. The locale and character set are determined by the settings in the Windows **Control Panel** that govern date and time formats, user-level display languages, system-level display languages, and system locale settings.

For details about the character set JP1/AO uses for communication, see [6.1.7 Character set used for communication by JP1/AO during plug-in execution](#).

## In UNIX

The locale setting applied during plug-in execution depends on the **Character Set Auto Judgment** setting on the **Remote Command** tab of the **Create Custom Plug-in** or **Edit Custom Plug-in** dialog box.

- If the **Enabled** check box is not selected in the **Character Set Auto Judgment** area  
Scripts are executed 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 be garbled and prevent the command from executing normally.
- If the **Enabled** check box is selected in the **Character Set Auto Judgment** area  
JP1/AO references the default locale of the connecting user and executes the script accordingly.  
When executing a script or command on an operation target device, JP1/AO sets the environment variable LC\_ALL and LANG to the default locale of the connecting user. It does not change the settings of LC\_XXXXX environment variables other than LC\_ALL.  
The locale assigned when executing a script or command is referenced in the order of priority as shown in the following table.

Table 6-4: Priority of locale settings referenced during plug-in execution

Priority	Environment variable
1	Value of LC_ALL
2	Value of LC_CTYPE
3	Value of LANG

If the script or command is encoded in a different character set from the one assigned at plug-in execution, the characters might be garbled and the script or command might not function properly. Note that the character set you can use in commands and command parameters depends on the operating system. For details, see [6.1.7 Character set used for communication by JP1/AO during plug-in execution](#).

## 6.1.7 Character set used for communication by JP1/AO during plug-in execution

The character set that is assigned at plug-in execution and used by JP1/AO for communication depends on the operating system of the device on which the operation is being performed.

Entries output to the task log and public log by the JP1/AO server are output in UTF-8. For this reason, characters taken from a character set that is incompatible with the operation target device, machine-dependent characters, and Unicode-dependent characters might be garbled when output to a log file.

The following describes the character sets assigned at plug-in execution according to the operating system of the operation target device.

### In Windows:

When executing a script or command on an operation target device, make sure that the locale and character set of the operation target device match those of the JP1/AO server. The locale and character set are determined by the settings in the **Control Panel**.

### In UNIX:

When executing a script or command on an operation target device, the character sets the JP1/AO server can use for communication are limited to those shown below as output by the `locale charmap` command. Note that the output of the `locale charmap` command is not case sensitive.

- EUC-JP
- eucjp
- ibm-943C
- ISO-8859-1
- MS932
- PCK
- Shift\_JIS
- UTF-8
- windows-31j

If the command returns a character set that is not one of those listed here, UTF-8 is assigned as the character set. Note that if the output of the `locale charmap` command is IBM-943, JP1/AO uses the ibm-943C character set for communication when executing the plug-in.

To find out which character set JP1/AO is using, use an SSH client or the `ssh` command to log in as the connection user, and then execute the `locale charmap` command. If you want to automatically change the character set when the connection user logs in, use a login script or other means to assign values to environment variables at login. You can change the character set at login by assigning a value to the `LC_ALL` or `LANG` environment variable.

If you want to assign a specific character set, see [6.1.8 Setting a specific character set during plug-in execution](#).

Note that if local execution function is enabled, the default character set for the System account or root user applies. In this case, the following processing is not performed:

- Character set auto judgment by the `locale charmap` command
- Character set setting by a character-set mapping file
- Character set setting by a connection-destination property file

## 6.1.8 Setting a specific character set during plug-in execution

If you want the JP1/AO server to use a specific character set for communication when performing an operation on a UNIX device, enter the appropriate setting in a character-set mapping file, or in the `terminal.charset` key of a connection-destination property file.

If you specify a character set in a character-set mapping file and in the `terminal.charset` key of a connection-destination property file, the character set is assigned in the order of priority shown in the following table.

Table 6-5: Priority of character set settings during plug-in execution

Priority	Setting
1	Character set specified in the <code>terminal.charset</code> key of the connection-destination property file
2	Character set specified in the character-set mapping file
3	Character set returned by the <code>locale charmap</code> command on the operation target device
4	UTF-8

---

## Related topics

- Connection-destination property file (connection-destination-name.properties) and Character-set mapping file (charsetMapping\_user.properties) in the JP1/Automatic Operation Configuration Guide
-

## 6.2 Creating and editing plug-in definition information

---

After you create, edit, or copy a plug-in, set the definition information.

### 6.2.1 Procedure for creating plug-ins

Users can create new plug-ins to meet a specific need.

#### To create a plug-in:

1. In the **Service Builder Home** window, from the **Custom Plug-in Actions** pull-down menu, select **Create**.
2. In the **Create Custom Plug-in** dialog box, enter the definition information for the plug-in and then click the **Save** button.

Figure 6-3: **Create Custom Plug-in** dialog box

The figure consists of three screenshots of the 'Create Custom Plug-in' dialog box, showing different tabs.

**General Tab:** This tab contains input fields for 'Key Name', 'Version' (01.00.00), 'Vendor ID', 'Display Name', 'Vendor Name', 'Description', and 'Tags'. There is also an 'Icons' section with a default icon and buttons for 'Restore Default Icon' and 'Change'.

**Property Tab:** This tab shows a table with columns: Key, Display Name, Description, Required, and Default Value. The table contains one row with the key 'plugin.destinationHost', display name 'Destination host', description 'For this property, specif...', required status 'true', and default value.

**Remote Command Tab:** This tab shows a 'Credential Type' section with radio buttons for 'Shared agentless setting' (selected) and 'Service input property'. Below this is a 'Platform' section with a large text area containing the instruction: 'Click [Add Platform] to add a remote OS command that will be run by the plug-in.' There is an 'Add Platform' button in the top right corner of the platform section.

**Tip**

You can also create a plug-in while editing a service template by clicking the **Create** button on the **Developing Plug-in** tab of the **Flow** tab of the **Service Builder Edit** window.

**Operation result**

A plug-in is created.

---

## Related topics

- [6.2.3 Items to set in plug-in definition information](#)
- 

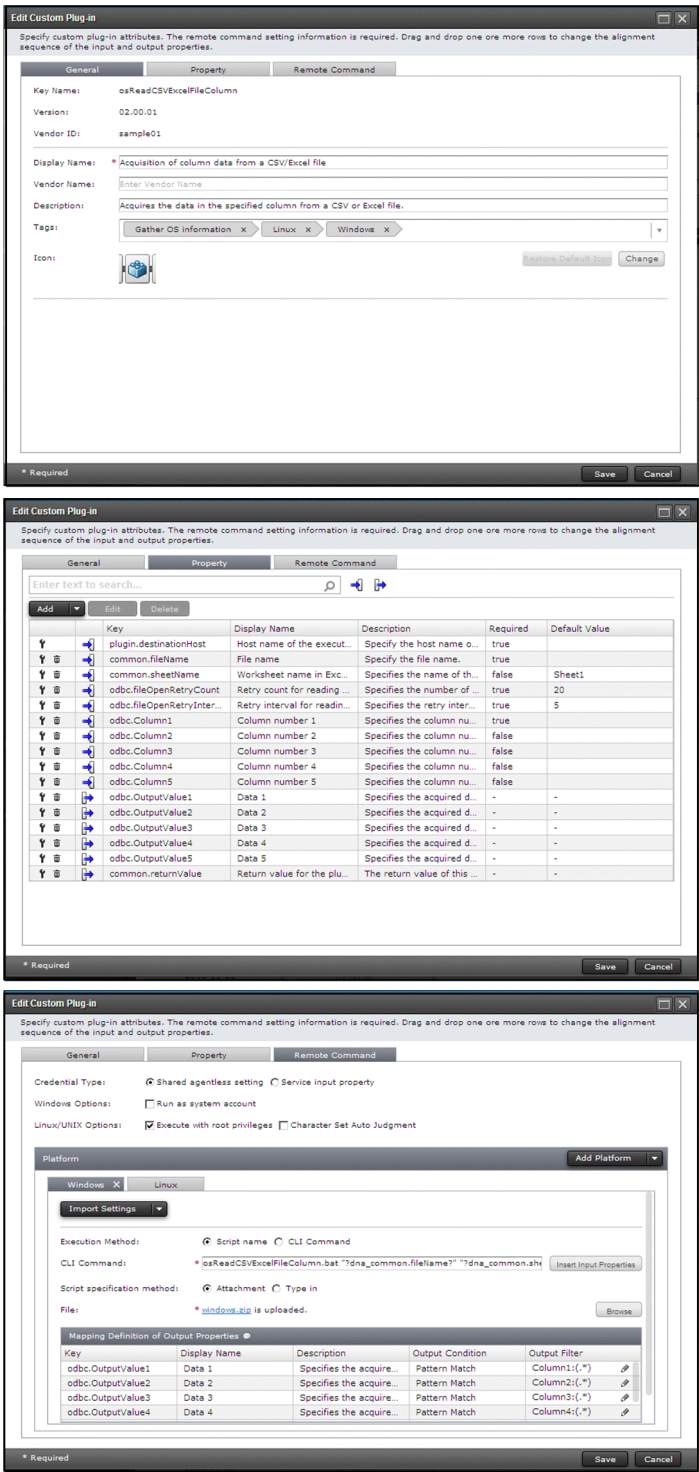
### 6.2.2 Procedure for editing plug-in definition information

You can edit the definition information for development plug-ins. When you want to edit a release plug-in, copy a development plug-in and then edit it.

#### To edit the definition information for a plug-in:

1. In the **Service Builder Home** window, from the **Custom Plug-in Actions** pull-down menu, select **Edit**.
2. In **Custom Plug-in List**, select the plug-in you want to edit, and then click the **Edit** button.
3. In the **Edit Custom Plug-in** dialog box, set the definition information of the plug-in, and then click the **Save** button.

Figure 6-4: Edit Custom Plug-in dialog box



## Operation result

The definition information of the plug-in is set.



## Important

If you are editing the properties of a plug-in that has been placed as a step in a flow, make sure that the settings of the step properties are appropriate. If there is an inconsistency between the settings for the plug-in properties and the step properties, an error will occur when you build or release the service template.

## Tip

You can also edit a plug-in while editing a service template by clicking the **Edit** button on the **Developing Plug-in** tab of the **Flow** tab of the **Service Builder Edit** window.

## Related topics

- [6.2.3 Items to set in plug-in definition information](#)

## 6.2.3 Items to set in plug-in definition information

The following table lists the items you can set on each tab when you create, edit, or copy a plug-in.

Table 6-6: Items to set in plug-in definition information (**General** tab)

Item	Description
<b>Key Name</b> <sup>#1</sup>	Specify the ID that identifies the plug-in.
<b>Version</b> <sup>#1</sup>	Specify the version number of the plug-in in aa.bb.cc format.
<b>Vendor ID</b> <sup>#1</sup>	Specify the ID that identifies the vendor who created the plug-in. Create a vendor ID that is unique within the system by specifying the domain name in reverse order from the top level as a period-separated value. For example, specify the vendor ID as com.xxxx or jp.co.yyyy. If you choose not to use domain names for vendor IDs, make sure that the vendor ID you specify is not being used for another vendor ID. Note that you cannot specify a vendor ID that begins with com.hitachi.software.dna.
<b>Display Name</b>	Specify the name of the plug-in.
<b>Vendor Name</b> <sup>#2</sup>	Specify the name of the vendor who created the plug-in.
<b>Description</b>	Specify the description for the plug-in.
<b>Tags</b>	Specify the tags you define for the plug-in. You can specify multiple tags. The total number of characters for all tags combined must be no more than 256, including the characters for all tag names set for the plug-in and the commas inserted between tags.
<b>Icon</b>	The icon set for the plug-in is displayed. Clicking the <b>Restore Default Icon</b> button changes the icon set for the plug-in back to the default. Clicking the <b>Change</b> button displays the dialog box where you can select the icon file to be uploaded and change the icon. For the icon, set the file in .png format (48 x 48 pixels).

#1

You cannot change the values for **Key Name**, **Version**, and **Vendor ID** after you create or copy a plug-in.

#2

If you omit specifying the value for this item, the value set for **Vendor ID** will be set for **Vendor Name**. Note that **Vendor Name** for a development plug-in displayed in the **Flow** tab of the **Service Builder Edit** window remains blank.

Table 6-7: Items to set in plug-in definition information (**Property** tab)

Item	Description	See
Input Properties <sup>#</sup>	You can define properties for storing input values necessary for executing the plug-in.	<a href="#">6.3 Setting plug-in properties</a>
Output Properties <sup>#</sup>	You can define properties for storing the execution results of the plug-in.	

#

You can define no more than 100 properties (as the total of input and output properties) for a plug-in.



### Tip

You can filter the displayed properties by clicking the icons for input and output properties.

Table 6-8: Items to set when a plug-in is created or edited (**Remote Command** tab)

Item	Description
<b>Credential Type</b>	Select the credential type for the plug-in from the following: <ul style="list-style-type: none"><li>• <b>Shared agentless setting</b> Select this option when you use the credential type set in the <b>Agentless Remote Connections</b> area when executing a service.</li><li>• <b>Service input property</b> Select this option when you specify authentication information by using an input property.</li></ul>
<b>Windows Options</b>	If you select the <b>Run as system account</b> check box, commands and scripts executed on the destination host are executed with the permissions of the System account.
<b>Linux/UNIX Options</b>	<ul style="list-style-type: none"><li>• <b>Execute with root privileges</b> Select the check box to run commands and scripts on the connection destination host as an executor with root privileges.</li><li>• <b>Character Set Auto Judgment</b> Select the check box to enable the Character Set Auto Judgment functionality.</li></ul>
<b>Platform</b>	Set a command or script that can be executed on the operation target host. Set this item for each OS. For details about the items you can set, see <a href="#">6.4 Editing platforms</a> .

## Related topics

- [6.2.4 Image files that can be set for component icons](#)
- [6.2.5 Plug-in credential types](#)

## 6.2.4 Image files that can be set for component icons

You can set any image for each component icon displayed in the **Flow** tab of the **Service Builder** window.

Set a plug-in icon when you create or edit a plug-in. For a service component, the service template icon set in the definition information of the service template is used as the component icon. If you do not set any icon, the default plug-in icon or default service template icon is set.

Figure 6-5: Default plug-in icon



Figure 6-6: Default service template icon



You can set an image file that satisfies the conditions below for an icon. If you specify a file that does not satisfy the conditions, an error occurs when the file is registered.

File format

PNG format

Image size

48 x 48 pixels

The file name of the image registered as an icon is changed to icon.png.

Note that you can select and re-register an image file that has already been registered. In this case, the existing registered file is deleted, and replaced with the re-registered file.

## 6.2.5 Plug-in credential types

The following properties are set automatically according to the option selected for **Credential Type** in the **Create Custom Plug-in** dialog box or the **Edit Custom Plug-in** dialog box:

When **Shared agentless setting** is selected for **Credential Type**

- plugin.destinationHost

When **Service input property** is selected for **Credential Type**

- plugin.destinationHost
- plugin.account
- plugin.password
- plugin.suPassword
- plugin.publicKeyAuthentication
- plugin.keyboardInteractiveAuthentication

---

### Related topics

- [6.3.6 Reserved plug-in properties for specifying execution-target hosts and authentication information](#)
-

## 6.3 Setting plug-in properties

---

After you create definition information for a plug-in, you can set the input properties and output properties for the plug-in.

### 6.3.1 Overview of plug-in properties

By defining plug-in properties, you can specify parameters necessary for executing plug-ins, or obtain the execution results of plug-ins. The following types of plug-in properties exist:

#### Input property

You can define input properties as properties used to store the input values necessary for executing plug-ins (for example, arguments used by the remote command, and operation target hosts).

#### Output property

You can define output properties as properties used to store the execution results of plug-ins (for example, the execution results of the remote command, such as the standard output or standard error output).

Plug-in properties are valid only in the plug-in for which the properties are defined.

The maximum size of an input property for a plug-in is 1,024 characters when a composite-type service property has not been mapped, or 30MB when a composite-type service property has been mapped. If the specified value exceeds the maximum value, the excess part of the value is truncated.

A plug-in property whose property key and intended use are predefined is called a *reserved plug-in property*. You need to define reserved plug-in properties as plug-in properties to specify the execution host of a remote command or authentication information.

---

#### Related topics

- [6.3.2 Procedure for adding plug-in properties](#)
  - [6.3.3 Procedure for editing plug-in properties](#)
  - [6.3.6 Reserved plug-in properties for specifying execution-target hosts and authentication information](#)
- 

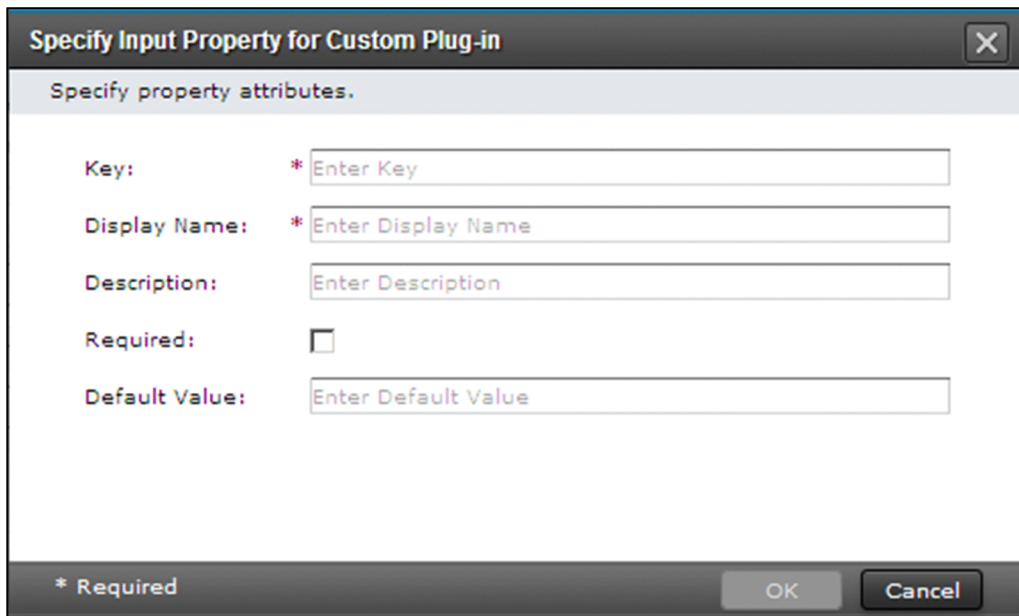
### 6.3.2 Procedure for adding plug-in properties

You can add plug-in properties when creating or editing plug-ins.

#### To add a plug-in property:

1. In the **Service Builder Home** window, perform operations for creating or editing a plug-in.
2. In the dialog box that appears, select the **Property** tab, and from the **Add** pull-down menu, select **Input Property** or **Output Property**.
3. In the dialog box that appears, enter the definition information for the plug-in property.

Figure 6-7: **Specify Input Property for Custom Plug-in** dialog box



The dialog box is titled "Specify Input Property for Custom Plug-in" and contains a section "Specify property attributes." with the following fields:

- Key:** \* Enter Key (required, text input)
- Display Name:** \* Enter Display Name (required, text input)
- Description:** Enter Description (text input)
- Required:** ☐ (checkbox)
- Default Value:** Enter Default Value (text input)

At the bottom, there is a legend "\* Required", and "OK" and "Cancel" buttons.

4. Click the **OK** button.

## Operation result

A plug-in property is added.

---

## Related topics

- [6.3.4 Items to set for plug-in input properties](#)
  - [6.3.5 Items to set for plug-in output properties](#)
  - [6.3.1 Overview of plug-in properties](#)
  - [6.2.1 Procedure for creating plug-ins](#)
  - [6.2.2 Procedure for editing plug-in definition information](#)
- 

## 6.3.3 Procedure for editing plug-in properties

You can edit plug-in properties when editing plug-ins.

### To edit a plug-in property:

1. In the **Service Builder Home** window, perform operations for editing a plug-in.
2. In the dialog box that appears, on the **Property** tab, select the row that corresponds to the property you want to edit, and then click the **Edit** button.
3. In the dialog box that appears, edit the definition information for the plug-in property.

Figure 6-8: **Edit Input Property for Custom Plug-in** dialog box

**Edit Input Property for Custom Plug-in**

Specify property attributes.

Key: \* Input

Display Name: \* Input Property

Description: Specifies the column number of the data to acquire.

Required: ☒

Default Value: Enter Default Value

\* Required

OK Cancel

4. Click the **OK** button.

**Operation result**

The definition information for the plug-in property is set.

**Related topics**

- [6.3.4 Items to set for plug-in input properties](#)
- [6.3.5 Items to set for plug-in output properties](#)
- [6.3.1 Overview of plug-in properties](#)
- [6.2.2 Procedure for editing plug-in definition information](#)

**6.3.4 Items to set for plug-in input properties**

In the **Specify Input Property for Custom Plug-in** dialog box or **Edit Input Property for Custom Plug-in** dialog box, you can set the items shown in the following table.

Table 6-9: Items to set for the definition information of plug-in input properties

Item	Description
Key	Specify the property key. You cannot change the value for this item when you have selected a reserved plug-in property.
Name	Specify the property name.
Description	Specify the description for the property.
Required	If you select the <b>Required</b> check box, entering a value for this item will be required. You cannot change the value for this item when you have selected the plugin.destinationHost, plugin.publicKeyAuthentication, or plugin.keyboardInteractiveAuthentication reserved plug-in property.
Default Value	Specify the default value for the property.

## 6.3.5 Items to set for plug-in output properties

In the **Specify Output Property for Custom Plug-in** dialog box or **Edit Output Property for Custom Plug-in** dialog box, you can set the items in the following table.

Table 6-10: Items to set for the definition information of plug-in output properties

Item	Description
Key	Specify the property key.
Name	Specify the property name.
Description	Specify the description for the property.

## 6.3.6 Reserved plug-in properties for specifying execution-target hosts and authentication information

A reserved plug-in property is a plug-in property whose property key and intended use are predefined in the JP1/AO system.

The names of reserved plug-in properties start with `plugin..` Reserved plug-in properties are created automatically in the **Input Properties** area of the following dialog boxes:

- **Create Custom Plug-in** dialog box
- **Edit Custom Plug-in** dialog box

Reserved plug-in properties are only valid in the context of the plug-in for which they are defined.

Although you can edit some aspects of a reserved plug-in property, other aspects such as the property key and whether certain parameters are mandatory cannot be changed. You cannot delete a reserved plug-in property.

### Reserved plug-in property for specifying execution-target hosts

The reserved plug-in property shown in the following table is automatically created to specify the execution-target host.

Table 6-11: Reserved plug-in property for specifying the execution-target host

Property key	Description
<code>plugin.destinationHost</code>	Specify the target host of an operation by IPv4 address, IPv6 address, or host name. You must specify a target host in a network configuration in which the JP1/AO server and the command execution environment can communicate directly with each other. However, if the OS on the JP1/AO server is Linux and the OS on the operation target device is Windows, you cannot specify an IPv6 address for the connection destination. You can specify a value from 1 to 256 characters long.

### Reserved plug-in property for specifying authentication information

As the credential type of a plug-in, for the **Credential Type** option on the **Remote Command** tab of the **Create Custom Plug-in** or **Edit Custom Plug-in** dialog box, select **Shared agentless setting** or **Service input property**.

## Shared agentless setting

Select this option to use the authentication information set in the **Agentless Remote Connections** area. When you select this option, JP1/AO uses the authentication information specified in the connection destination definition for WMI, SSH, or Telnet connections, depending on the operation-target host.

## Service input property

Select this option to use the authentication information specified in a property.

When you select **Service input property**, the reserved plug-in properties shown in the following table are automatically created in the list of input properties in the **Create Custom Plug-in** dialog box or **Edit Custom Plug-in** dialog box. Note that if local execution function is enabled and the execution host is the local host, these settings are ignored.

Table 6-12: Reserved plug-in properties for specifying authentication information

Property key	Description
plugin.account	Specify the user ID for logging in to the target host in 1 to 256 characters.
plugin.password	Specify the password for logging in to the target host in 1 to 256 characters. If true is specified for the plugin.publicKeyAuthentication reserved plug-in property, JP1/AO ignores the value set for the plugin.password property.
plugin.suPassword	Specify the password of the root account used to log in to a target host in a UNIX environment, using 1 to 256 characters. The root password you specify is ignored in the following circumstances: <ul style="list-style-type: none"><li>• The target host is running Windows.</li><li>• The <b>Execute with root privileges</b> check box is not selected in the <b>Create Custom Plug-in</b> dialog box or the <b>Edit Custom Plug-in</b> dialog box.</li></ul>
plugin.publicKeyAuthentication	This property specifies whether to use public key authentication for SSH connections to target hosts in UNIX environments. If you do not specify a value, false is the default value. <ul style="list-style-type: none"><li>• true Specify this value to use public key authentication.</li><li>• false Specify this value to not use public key authentication.<sup>#</sup></li></ul>
plugin.keyboardInteractiveAuthentication	This property specifies whether to use keyboard interactive authentication for SSH connections to target hosts in UNIX environments. If you do not specify a value, false is the default value. Note that the value for plugin.keyboardInteractiveAuthentication is enabled only when false is specified for plugin.publicKeyAuthentication. If true is specified for plugin.publicKeyAuthentication, public key authentication is set even if true is specified for plugin.keyboardInteractiveAuthentication. <ul style="list-style-type: none"><li>• true Specify this value to use keyboard interactive authentication.</li><li>• false Specify this value to not use keyboard interactive authentication.<sup>#</sup></li></ul>

#

If you specify false for both plugin.publicKeyAuthentication and plugin.keyboardInteractiveAuthentication properties, password authentication is set.

## 6.3.7 Procedure for deleting plug-in properties

You can delete plug-in properties when editing plug-ins.





### Tip

You cannot delete reserved plug-in properties.

## To delete a plug-in property:

1. In the **Service Builder Home** window, perform operations for editing a plug-in.
2. In the dialog box that appears, select the row that corresponds to the property you want to delete on the **Property** tab, and then click the trash can icon.

## Operation result

A plug-in property is deleted.

---

## Related topics

- [6.2.2 Procedure for editing plug-in definition information](#)
  - [6.3.6 Reserved plug-in properties for specifying execution-target hosts and authentication information](#)
-

## 6.4 Editing platforms

You can set the commands or scripts that plug-ins execute on the operation-target hosts.

### 6.4.1 Procedure for editing platforms

You can assign a platform to a plug-in. You can set different platforms for each operating system.

#### To edit a platform:

1. In the **Service Builder Home** window, perform operations for creating or editing a plug-in.
2. In the dialog box that appears, click the **Remote Command** tab, and edit the items in the **Platform** area.

Figure 6-9: **Platform** (when **Script** is selected for the execution method)

The screenshot shows the 'Edit Custom Plug-in' dialog box with the 'Remote Command' tab selected. The 'Platform' section is expanded, showing the 'Execution Method' set to 'Script name'. The 'Script specification method' is set to 'Attachment'. Below this is a table for 'Mapping Definition of Output Properties' with columns: Key, Display Name, Description, Output Condition, and Output Filter. A message at the bottom of the table says 'Select the Property tab, and then click [Add] to add an Output Property.'

Key	Display Name	Description	Output Condition	Output Filter
-----	--------------	-------------	------------------	---------------

3. Click the **Save** button.

### Operation result

The platform to be executed by the OS on the operation-target host is set.

#### Related topics

- [6.4.2 Items to set for platforms](#)
- [6.2.1 Procedure for creating plug-ins](#)
- [6.2.2 Procedure for editing plug-in definition information](#)

## 6.4.2 Items to set for platforms

In the **Platform** area, select the operating system on which the remote command will be executed from the **Add Platform** pull-down menu.

If you select another OS in the **Import Settings** pull-down menu, the definition information of the remote command targeted to the selected OS is loaded. You can select the following OSs:

- Windows
- AIX
- HP-UX
- Linux
- Solaris

Note that you need to upload the attached file for the specified OS after the definition information is loaded because the attached file is not loaded automatically.

The following table lists the items you can set on the tab for the selected operating system.

Table 6-13: Items to set on tab for selected OS

Item	Description
<b>Execution Method</b>	Select the execution method of the remote command. <ul style="list-style-type: none"><li>• <b>Script</b> Select this option when you want to set a script created by a user.</li><li>• <b>CLI Command</b> Select this option when you want to set a command stored in the operation-target device.</li></ul>
<b>CLI Command</b>	Enter the command line to be executed. If you click the <b>Insert Input Property</b> button, all input properties defined for the plug-in are displayed. When you select a property or properties to insert and click <b>OK</b> , the property or properties are input in the format <code>?dna_property-key?</code> . You can select multiple properties by selecting the check boxes beside the property keys.
<b>Script specification method</b>	If you select <b>Script</b> for the <b>Execution Method</b> radio button, select the method for specifying the script. <ul style="list-style-type: none"><li>• <b>Attachment</b> Select this option when you want to attach the script.</li><li>• <b>Type in</b> Select this option when you want to directly enter the script.</li></ul>
<b>File</b>	If you select <b>Attachment</b> for the <b>Script specification method</b> radio button, the name of the attached file is displayed. If you click the <b>Browse</b> button, a dialog box appears in which you can select the script file to be attached.
<b>File Name</b>	If you select <b>Type in</b> for the <b>Script specification method</b> radio button, enter the file name of the script entered in the <b>Script</b> text box. The script entered in the <b>Script</b> text box is uploaded with the file name set for this item.
<b>Script</b>	If you select <b>Type in</b> for the <b>Script specification method</b> radio button, enter the script you want to execute on the operation-target device.
<b>Mapping Definition of Output Properties</b>	Displays a list of mapping definitions of output properties. If you want to store part of the standard output and standard error output of a commands or script in an output property, click the pencil icon beside the property in which you want to store the information. This allows you to edit the output filter.
<b>Details</b>	<b>Execution Directory</b> Enter the directory where the command or script is to be executed on the target device.

Item	Description	
Details	Environment Variables	Displays a list of environment variables required to execute commands or scripts. You can add, edit, or delete environment variables.

## Related topics

- [6.4.3 Procedure for setting commands](#)
- [6.4.4 Method for specifying scripts](#)
- [6.4.12 Procedure for mapping standard output and standard error output to output properties](#)
- [6.4.14 Specifying \*\*Execution Directory\*\*](#)
- [6.4.15 Procedure for adding and editing environment variables](#)

## 6.4.3 Procedure for setting commands

Set the command that is appropriate for the operating system on the operation target host (Windows, AIX, HP-UX, Linux, or Solaris). When creating a plug-in that is compatible with multiple operating systems, set a command for each operating system.

The commands you set must be scripts or commands that are present on and executable by the operation target device.

The commands executed in plug-ins can have return values in the range from 0 to 63.

Plug-ins and service templates must be designed in such a way that standard output and standard error output produce less than 100 KB of data. When the standard output or standard error output of a plug-in exceeds 100 KB, the command is immediately killed and the plug-in terminates with an error. In this scenario, the results of execution of the command cannot be guaranteed.



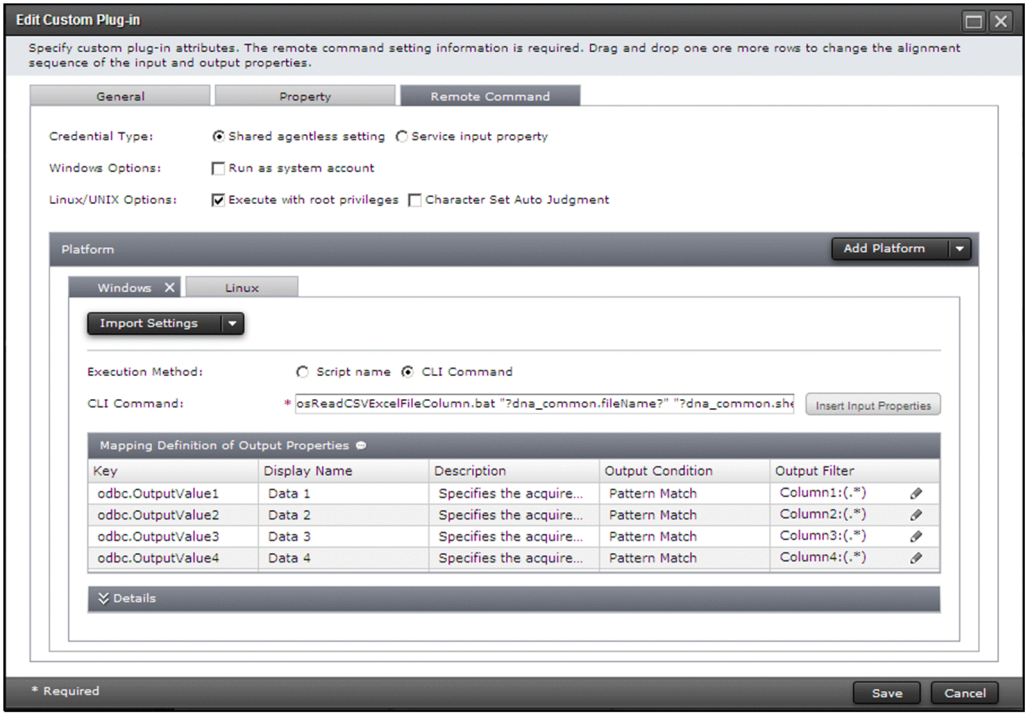
### Important

Interactive commands and script that seek user input and commands that do not end automatically using a GUI display or the like cannot be executed.

## To set a command:

1. Click the **Remote Command** tab in the **Create Custom Plug-in** or **Edit Custom Plug-in** dialog box.
2. In the **Execution Method** area of the **Platform**, click **CLI Command**.
3. Enter the command line in **CLI Command**, and then click the **Save** button.

Figure 6-10: **Platform** (when **CLI Command** is selected for the execution method)



Related topics

- 6.4.7 Specifying commands in the **CLI Command** text box
- 6.4.8 Procedure for using the return value of a command or script as a flow branching condition (for values outside the 0 to 63 range)

6.4.4 Method for specifying scripts

You can set a script for each OS to create plug-ins that match the OS on the operation target device. You can define a script or command that exists and can be executed on the operation target device.

You can re-register the script file that has already been registered. The existing file is overwritten when the file is re-registered.

Difference between the methods for specifying scripts

You can specify a script by attaching a script file that has been created, or by directly entering the script. The differences between these two methods is described below.

When **Attachment** is selected:

If you use zip format, you can register a script composed of multiple files or multiple folders. Note that compressed files with zip extension are renamed when you save plug-ins in the JP1/AO server. The file name is changed to windows.zip, aix.zip, hpux.zip, linux.zip, or solaris.zip, depending on the OS. You can download the file you registered by selecting **Attachment**, to the terminal on which you perform JP1/AO operations via a Web browser. If you directly enter a script, you cannot download the script file you defined.

When **Type in** is selected:

In the **Edit Remote Command** dialog box, directly enter the content of the script you want to execute on the operation target device. You can define a script or command that exists and can be executed on the operation target device. In this case, you can define only one script file. The character set and linefeed code for the script to be saved are fixed depending on the OS on the operation target device.

Therefore, if you want to register a script file composed of multiple files or if you want to set any character set and linefeed code you want, select **Attachment**.

The following table describes the differences in character sets and linefeed codes that are set and how to register files, between the settings for **Script specification method**.

Table 6-14: Difference between the methods of setting scripts

Item	When Attachment is selected		When Type in is selected	
	Windows	AIX, HP-UX, Linux, and Solaris	Windows	AIX, HP-UX, Linux, and Solaris
Registration of a file	Can be registered.		Can be registered.	
Registration of multiple files	Can be registered.		Cannot be registered.	
Character set for the script to be saved	The character set for the registration file is set.		The default character set for the OS on the JP1/AO server	UTF-8
Linefeed code for the script to be saved	The linefeed code for the registration file is set.		CR+LF	LF

## Notes on creating script files

Note the following when you create a script file:

- Use ASCII characters for a script file. You cannot use the following characters:
  - Control characters ('`\u0000`' to '`\u001F`', or '`\u007F`' to '`\u009F`')
  - Question marks (?), asterisks (\*), double quotation marks ("), right angle brackets (>), left angle brackets (<), vertical bars (|), and colons (:)
- You cannot use multi-byte characters for a file name.
- The scripts executed in plug-ins can have return values in the range from 0 to 63.
- Plug-ins and service templates must be designed in such a way that standard output and standard error output produce less than 100 KB of data. When the standard output or standard error output of a plug-in exceeds 100 KB, the script is immediately stopped and the plug-in terminates with an error. In this scenario, the results of execution of the script cannot be guaranteed.
- The locale used when a script is executed differs, depending on the OS on the operation target device. For details about the locale set for the operation target device, see [6.1.6 Locale set for operation target devices during plug-in execution](#).
- Interactive commands and script that seek user input and commands that do not end automatically using a GUI display or the like cannot be executed.



### Tip

For details about the setting of return values when a command or script is executed, see [6.4.9 Return values of content plug-ins](#).

## Related topics

- 6.4.5 Procedure for setting scripts (when attaching created scripts)
- 6.4.6 Procedure for setting scripts (when directly entering scripts)
- 6.4.8 Procedure for using the return value of a command or script as a flow branching condition (for values outside the 0 to 63 range)
- 6.4.10 Relationship of command and script return values to the return values of plug-ins and steps

## 6.4.5 Procedure for setting scripts (when attaching created scripts)

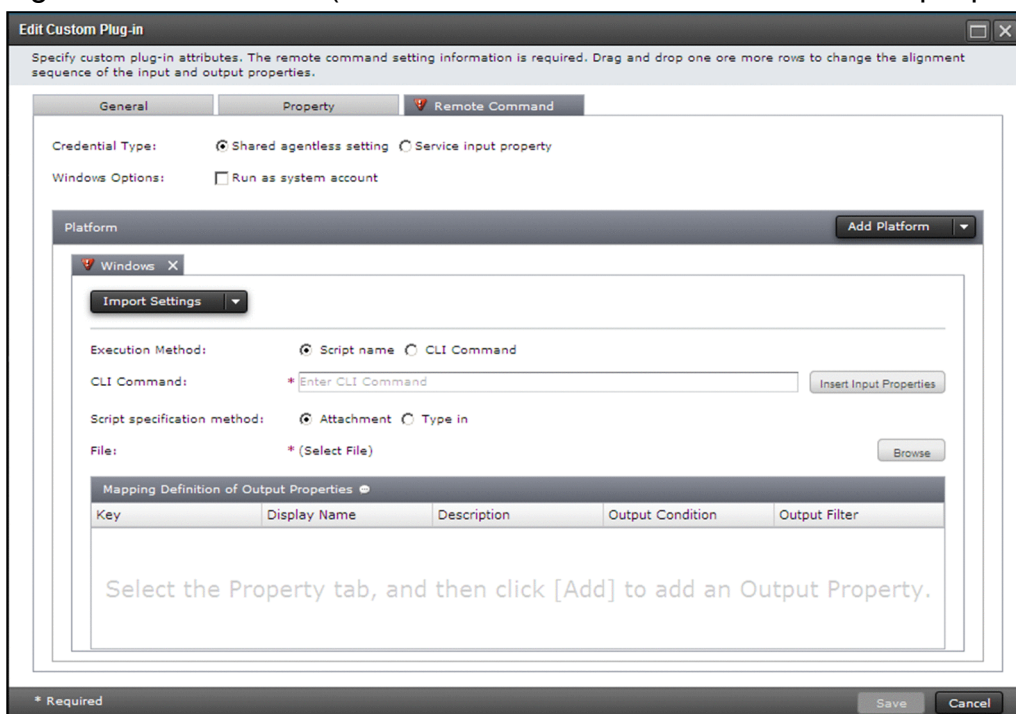
You can set the script to be executed on the operation target host, by specifying an attached file.

### To attach a script that has already been created:

1. Click the **Remote Command** tab on the **Create Custom Plug-in** or **Edit Custom Plug-in** dialog box.
2. In the **Execution Method** part of the **Platform** area, select the **Script** option.
3. In the **CLI Command** text box, enter the command that executes the script.

Although only one script file can be registered, you can register a script that consists of several files and folders in a hierarchical structure by compressing the files into a zip archive. If the script is a single file, specify the file name in the **CLI Command** text box. If the script is a zip archive containing multiple files, specify a relative path whose current directory is the location where the archive will be extracted.

Figure 6-11: **Platform** (when **Attachment** is selected for the script specification method)



4. In the **Script specification method** area, select the **Attachment** option.
5. In the **File** area, click **Browse** and register the script.

## Operation result

A script file is set.



---

## Related topics

- [6.4.4 Method for specifying scripts](#)
  - [6.4.7 Specifying commands in the CLI Command text box](#)
- 

## 6.4.6 Procedure for setting scripts (when directly entering scripts)

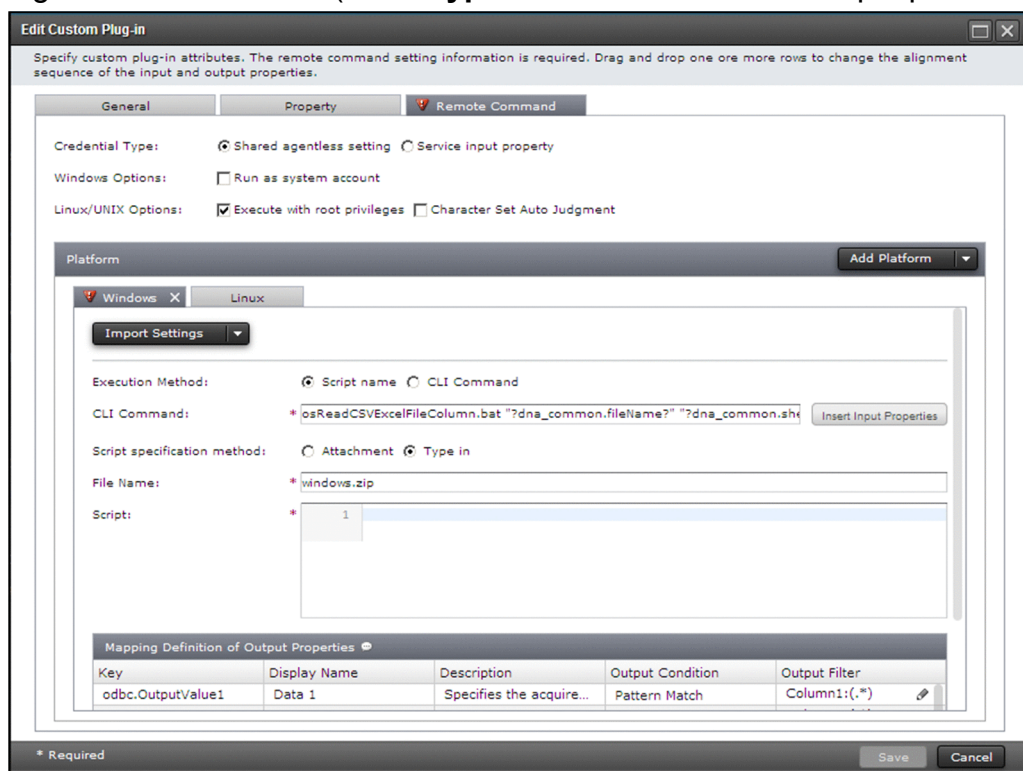
You can set the script to be executed on the operation target host, by directly entering the script.

### To directly enter a script:

1. Click the **Remote Command** tab in the **Create Custom Plug-in** or **Edit Custom Plug-in** dialog box.
2. In the **Execution Method** area in the **Platform**, select the **Script** option.
3. In the **CLI Command** text box, enter the command that executes the script.
4. In the **Script specification method** area, select the **Type in** option.
5. Enter values in the **File Name** field and the **Script** text box.

In the **File Name** field, enter the name of the file in which to store the code entered in the **Script** text box. In the **Script** text box, enter the script code.

Figure 6-12: **Platform** (when **Type in** is selected for the script specification method)



## Operation result

A script file is set.

---

## Related topics

- [6.4.4 Method for specifying scripts](#)



## 6.4.7 Specifying commands in the CLI Command text box

The information you can enter in the **CLI Command** text box depends on the option specified for **Execution Mode** and whether you choose to specify the value of an input property as an argument of the command you are executing.

In the **CLI Command** text box, you can specify characters other than control characters (\u0000 to \u001F and \u007F to \u009F). However, JP1/AO does not check the validity of the command entered in the **CLI Command** text box. Therefore, enter a command for which validation has been completed.

Note that special characters specified in the **CLI Command** text box, such as those used to indicate environment variables, will not be escaped. However, depending on the selection of **Operating System**, the following characters are automatically escaped when the values of mapped input properties are passed to the command line:

- In Windows: %
- In UNIX: \$, `, \, "

When mapping an input property as a command line argument, enclose the value of the argument in double quotation marks (for example, "?dna\_property-key-of-plug-in-property?"). When executing a PowerShell script, you can enclose the value in double or single quotation marks.

If **Operating System** is Windows and the value of an input property contains a double-quotation mark, an error will occur when the plug-in is executed.

### When Script is specified for Execution Method

Create the script to be executed on the target device, and enter the command that calls the script in the **CLI Command** text box. If the script is a single file, specify the file name. If the script is a zip archive containing multiple files, specify a relative path whose current directory is the location where the archive will be extracted.

The script is copied to a temporary folder under the folder specified in **Execution Directory**.

If **Operating System** is AIX, HP-UX, Linux, or Solaris, the command line is automatically prefixed with ./ when the command is executed. You do not need to manually add the prefix. If you specify ./ in the command line, the script file appears after ./ but still works normally. Special characters, such as those used to indicate environment variables in the command line, are not escaped.

The following shows how to enter information in the **CLI Command** text box when the **Script** option is specified for **Execution Method**:

Example of specifying information in the **CLI Command** text box

```
cmd.exe /q /c "AAA.bat bbb ccc"
```

Example of script file (AAA.bat) contents

```
@xxx.exe %~1
@yyy.exe %~2
```

If the value in **Operating System** is Windows, the command is converted to a batch file and executed on the operation target device. For this reason, the results of the command might differ from those of the same command executed at the command prompt.

## When CLI Command is specified for Execution Method

Enter the command to be executed on the operation target device directly in the **CLI Command** text box. You do not need to create a script.

The following shows how to enter information in the **CLI Command** text box when the **CLI Command** option is specified for **Execution Method**:

Example of specifying information in the **CLI Command** text box

```
zzz.exe aaa bbb
```

## When specifying input property values as command arguments

To specify the value of an input property in an argument of a command, specify *?dna\_property-key-of-plug-in-property?* in the **CLI Command** text box.

Example of specifying information in the **CLI Command** text box

```
scriptA.sh -xx ?dna_input01? -yy ?dna_input02?
```

In this example, *?dna\_input01?* is replaced with the value of the plug-in property *input01*, and *?dna\_input02?* is replaced with the value of the plug-in property *input02*.

## When specifying a non-standard script

JP1/AO executes scripts using *cmd.exe* when the operation target device is running Windows, and the user's login shell when the device is running UNIX. If you want to run a non-standard script, you need to define the instructions required to run the executable file that implements the script.

The following shows an example of running a PowerShell script from the command prompt and establishing a connection with vCenter:

Example of specifying information in the **CLI Command** text box

```
powershell -executionPolicy RemoteSigned -command ".\vsphereConnectChallenge.ps1  
'?dna_vCenterServerName?' '?dna_userName?' '?dna_password?' '?dna_portNumber?' '?  
dna_protocol?'; exit $LASTEXITCODE" 2>&1
```

- PowerShell cannot execute scripts by default. By specifying `powershell -executionPolicy RemoteSigned` in the command line, you can execute a local PowerShell script on an operation target device of JP1/AO.
- Here, *?dna\_property-key?* is a variable replaced with the value of a property. Enclose *?dna\_property-key?* with double or single-quotation marks.<sup>#</sup>

This allows the properties specified in the **CLI Command** text box to be passed to the shell even if the property has a null value.

<sup>#</sup> If you enclose a property with double-quotation marks (") in PowerShell and that property has a null value, PowerShell skips the property. If you enclose it in single-quotation marks ('), the property is interpreted as a null value and not skipped. By avoiding double-quotation marks ("), you can ensure that the script is executed as originally defined in terms of the order and content of arguments.

---

## Related topics

- [6.4.9 Return values of content plug-ins](#)
- [6.4.10 Relationship of command and script return values to the return values of plug-ins and steps](#)
- [6.4.11 Information output to standard output by plug-ins](#)

## 6.4.8 Procedure for using the return value of a command or script as a flow branching condition (for values outside the 0 to 63 range)

When a command or script executed in a plug-in returns a value outside the 0 to 63 range, you can use the return value as the branch condition for a flow by following the procedure below.

### To use the return value of a command or script as the branch condition of a flow:

1. When creating or editing a plug-in, specify the **Script** option for **Execution Mode** in the **Platform**.
2. Create a script that outputs the return value of the command it executes to standard output.
3. In the **Edit Output Filter** dialog box, enter a regular expression that assigns the return value of the command or script output to standard output to an output property of the plug-in.
4. Configure output property mapping so that the value of the plug-in output property assigned in step 3 is assigned to a service property (variable).
5. Configure a branch by property value plug-in (which is a basic plug-in) to judge the value of the service property (variable) assigned in step 4.

### Operation result

The return value is set as the branch condition for a flow.

### Related topics

- [6.4.10 Relationship of command and script return values to the return values of plug-ins and steps](#)
- branch by property value plug-in in the JP1/Automatic Operation Service Template Reference

## 6.4.9 Return values of content plug-ins

The following table lists the return values of content plug-ins.

Table 6-15: Return values of content plug-ins

Return value	Description
0 to 63 <sup>#</sup>	The meaning of the return value differs between content plug-ins.
64	A command executed in the content plug-in terminated with a return value outside the 0 to 63 range.
65	The connection to the JP1/AO server failed. For example, the JP1/AO server might have stopped during plug-in execution.
66	The following user is mapped to the JP1 user: <ul style="list-style-type: none"><li>• A user who does not belong to the Administrators group</li><li>• A user with UAC enabled who is not the built-in Administrator of the Administrators group</li></ul>
68	The system cannot find information for the applicable job execution ID.
70	The connection to the remote host failed.

Return value	Description
71	One of the following reasons applies: <ul style="list-style-type: none"> <li>• An attempt to call a command failed.</li> <li>• An attempt to move to the execution directory failed.</li> <li>• An attempt to set an environment variable failed.</li> </ul>
72	One of the following reasons applies: <ul style="list-style-type: none"> <li>• An attempt to acquire the execution status of a command failed.</li> <li>• The total data output to the standard output and standard error output exceeds 100KB.</li> </ul>
73	File transfer failed.
74	File deletion failed.
76	The connection timed out.
77	The host name of the remote host could not be resolved.
78	Authentication with the remote host failed for one of the following reasons: <ul style="list-style-type: none"> <li>• Password authentication failed.</li> <li>• Public key authentication has not been set up on the operation target device.</li> <li>• In public key authentication, the private key does not match the pass phrase.</li> <li>• In public key authentication, the private key does not correspond to the public key registered in the operation target device.</li> <li>• In public key authentication, an invalid private key was used.</li> <li>• Keyboard interactive authentication failed.</li> </ul>
80	Task execution has stopped.
81	The plug-in was called in an invalid status.
83	The environment of the JP1/AO server is corrupted.
84	Information about the specified plug-in could not be obtained.
86	The specified property value is invalid.
127	An unspecified error has occurred.

#

For plug-ins created in the **Service Builder Home** window, the return values are the same as for the command or script the plug-in is executing. If the plug-in is a content plug-in provided by JP1/AO, see the description of the content plug-in in the manual *JP1/Automatic Operation Service Template Reference*.

## Related topics

- [6.4.10 Relationship of command and script return values to the return values of plug-ins and steps](#)

## 6.4.10 Relationship of command and script return values to the return values of plug-ins and steps

In most circumstances, the return value of a command or script serves as the return value of the plug-in. When content plug-ins and general command plug-ins execute commands and scripts on connection destinations, the return value of the command or script is used as the return value of the plug-in. This does not apply to the terminal command plug-in, which does not set the return value of the command or script as the return value of the plug-in.

You can use values in the range from 0 to 63 as the return value of a command or script. If the command or script returns a value outside this range, the plug-in returns 64, indicating that the returned value was outside the 0 to 63 range. If the command or script could not be executed, the plug-in returns a value of 65 or higher that indicates the cause of the failure.

Although the return value of the plug-in is generally the return value of the step, the values might differ in certain circumstances, such as an error occurring during plug-in execution. In this case, take the appropriate action based on the return value of the step.

---

## Related topics

- [6.4.8 Procedure for using the return value of a command or script as a flow branching condition \(for values outside the 0 to 63 range\)](#)
- 

### 6.4.11 Information output to standard output by plug-ins

The standard output and standard error output of the commands and scripts specified in the **CLI Command** text box serve as the standard output of the plug-in.

However, plug-ins and service templates must be designed so that the standard output of the plug-in does not exceed 100 KB. If the standard output of a plug-in exceeds this size, the command is immediately forcibly terminated and the plug-in ends in an error. In this situation, the results of execution of the command cannot be guaranteed.

The size of standard output includes the data added by JP1/AO. For this reason, you need to include some leeway over the standard output and standard error output of the plug-in when estimating the standard output of a plug-in.

#### Size of plug-in standard output (when the operation target device is running UNIX)

Number of linefeed codes (LF) × bytes

When the operation target device is running UNIX, the carriage return character CR (0x0d) is replaced with the linefeed character LF (0x0a) in standard output and standard error output. LF (0x0a) is appended to the end of standard output and standard error output if the last character is not a linefeed character (CR/LF/CR+LF).

- LF (0x0a) is left unchanged.
- CR (0x0d) is replaced with LF (0x0a).
- CR+LF (0x0d0a) is replaced with LF+LF (0x0a0a).

---

## Related topics

- [6.4.12 Procedure for mapping standard output and standard error output to output properties](#)
- 

### 6.4.12 Procedure for mapping standard output and standard error output to output properties

You can map the standard output and standard error output of a command or script to an output property.

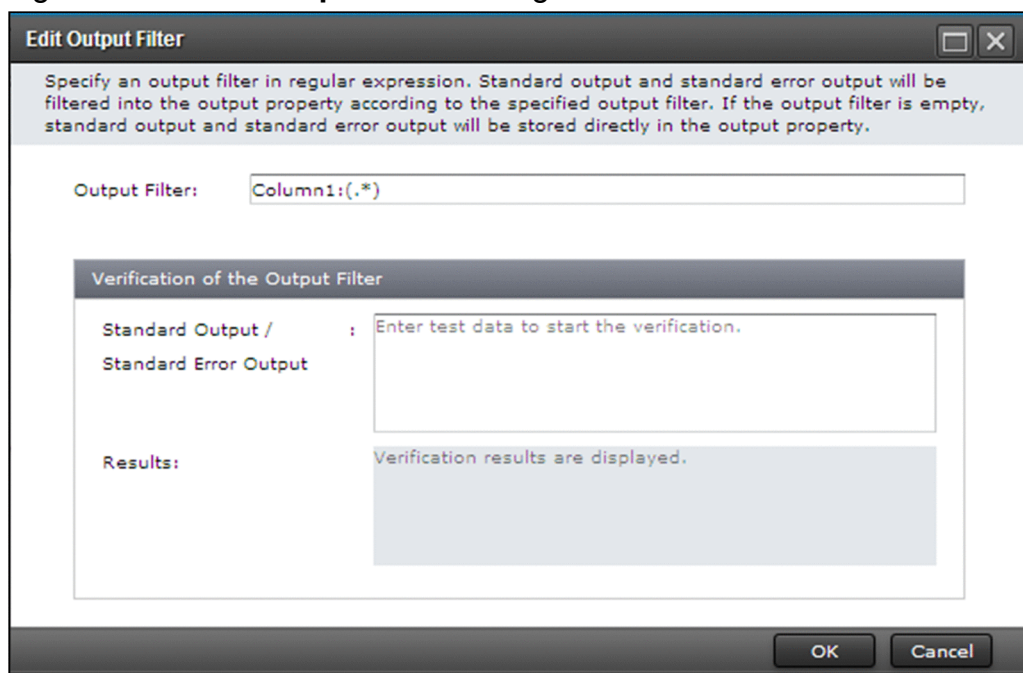
By default, the output filters are blank for properties in the list of mapping definition of output properties. This means that the entire standard output and standard error output of commands and scripts is stored in output properties. To

extract values from standard output and standard error output and map them to output properties, define regular expressions in the **Edit Output Filter** dialog box.

## To map standard output or standard error output to an output property:

1. Click the **Remote Command** tab in the **Create Custom Plug-in** or **Edit Custom Plug-in** dialog box.
2. Select a property in the list of mapping definitions of output properties, and then click the pencil icon.  
Only output properties that you registered when setting the plug-in properties that appear in the list of mapping definitions of output properties.
3. In the **Edit Output Filter** dialog box, enter a PCRE-compliant regular expression for the output filter, and then click the **OK** button.

Figure 6-13: **Edit Output Filter** dialog box



### Tip

You can also edit the filter by entering the new filter directly in the text box for the output filter.

4. You can verify the output filter you set by entering a sample of standard output and standard error output in the **Verification of the Output Filter > Standard Output / Standard Error Output** text box. The results of the test appear in **Results**.

## Operation result

The value extracted by the output filter is displayed in the **Results** area.

## Related topics

- [6.4.13 Specifying Output Filter](#)

## 6.4.13 Specifying Output Filter

This section describes how to store the standard output of a command or script in an output property.

By defining a PCRE-compliant regular expression in the **Output Filter** field, you can extract character strings from the standard output and standard error output of a command or script, and store them in the output property of a plug-in.

### Important

- If you specify multiple groups in the regular expression, only values that match the first group are stored in the output property of the plug-in.
- If the regular expression applies to multiple value ranges, only the first range of values is stored in the output property of the plug-in. Multiple value ranges cannot be stored in an output property.

The following describes how to specify a regular expression that stores the standard output of a command or script in an output property:

- **Key:** output01
- **Output Filter:** DATE= ( . \* )

When you specify an output filter in this way, the value immediately following DATE= in standard output is stored in the output property output01.

To store the return value of a script in an output property, define the plug-in and create a script as follows:

- In the **Platform** , specify the **Script** option for **Execution Method**.
- Create a script whose standard output displays the return value of the command or script in a format that is filtered by the regular expression specified in the output filter.

## 6.4.14 Specifying Execution Directory

This section describes how to specify the absolute path to the folder in which scripts or commands are executed. For the execution directory, specify characters that can be used with the commands of the OS on the JP1/AO server and of the OS on the operation target device.

Do not enclose the path to the execution directory by double quotation marks (") or single quotation marks (') even if the path contains a space character. If you enclose the path by quotation marks, execution of the plug-in fails. Also, you must create the execution directory on the operation target host in advance, and if necessary, change the permissions. At least, you need to grant necessary permissions to users who execute plug-ins.

If the OS on the operation target host is Windows, create the execution directory on the same drive with the working folder. If the execution directory is not correctly created in advance, execution of plug-ins might fail.

You can specify the execution directory in **Execution Directory**, or in the user-specified properties file (config\_user.properties) or connection-destination property file (*connection-destination-name.properties*). The execution directory is set according to the following order of priority:

1. The value specified in Execution Directory
2. The value specified for the common.executionDirectory key in the connection-destination property file (*connection-destination-name.properties*)



3. The value specified for the plugin.remoteCommand.executionDirectory.wmi key (when the operation target host is running Windows) or the plugin.remoteCommand.executionDirectory.ssh key (when the operation target host is running UNIX) in the user-specified properties file (config\_user.properties).
4. The value of the %TEMP% environment variable (when the operation target host is running Windows) or /tmp (when the operation target host is running UNIX)

Note that the behavior of JP1/AO differs as follows, depending on the value specified for **Execution Method**.

### When Script is specified for Execution Method

A script is copied to the temporary folder that has a unique name and is specified in **Execution Directory**. The script is executed in the temporary folder. Note that the copied script and temporary folder are deleted after execution of the script finishes.

### When CLI Command is specified for Execution Method

The command line is executed by using the directory specified in **Execution Directory**.

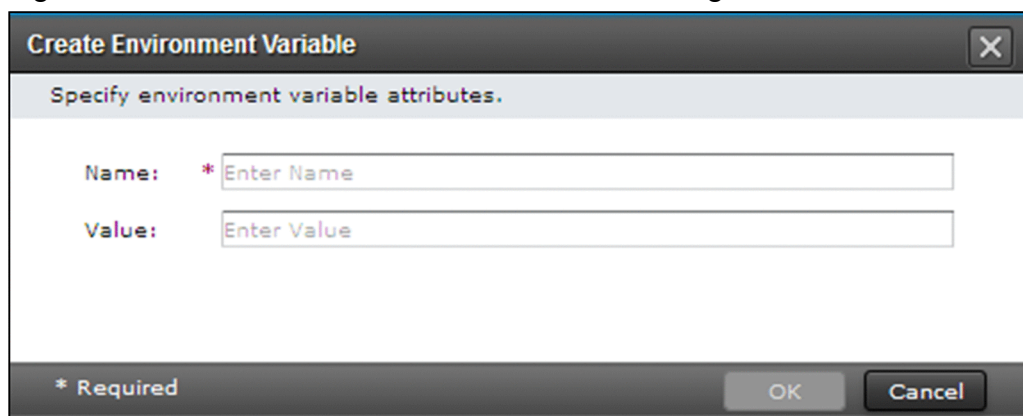
## 6.4.15 Procedure for adding and editing environment variables

You can add environment variables to remote commands, or edit existing environment variables.

### To add or edit an environment variable:

1. Click the **Remote Command** tab in the **Create Custom Plug-in** or **Edit Custom Plug-in** dialog box.
2. In the **Platform** area, under **Details > Environment Variables**, click the **Add** button. To edit an existing environment variable, select the environment variable and click the **Edit** icon.

Figure 6-14: **Create Environment Variable** dialog box



3. In the dialog box that appears, set the name and value for the environment variable, and then click the **OK** button.

### Operation result

The environment variable is set.

---

### Related topics

- [6.4.1 Procedure for editing platforms](#)
  - [6.4.16 Procedure for deleting environment variables](#)
-



## 6.4.16 Procedure for deleting environment variables

You can delete environment variables defined for remote commands.

### To delete an environment variable:

1. Click the **Remote Command** tab in the **Create Custom Plug-in** or **Edit Custom Plug-in** dialog box.
2. In the **Platform** dialog box, under **Details > Environment Variables**, select the environment variable you want to delete, and then click the **Delete** icon.
3. A dialog box (confirmation screen) appears asking you to confirm that you want to delete the environment variable. Click **OK**.

### Operation result

The environment variable is deleted.

## 6.5 Using resource files to set plug-in display information

This section describes how to assign resource files to plug-ins, and define the information to be displayed in windows for each Web browser locale. For example, you can display plug-in names in the language that is appropriate for the locale of the Web browser.

### 6.5.1 Procedure for setting plug-in resource files

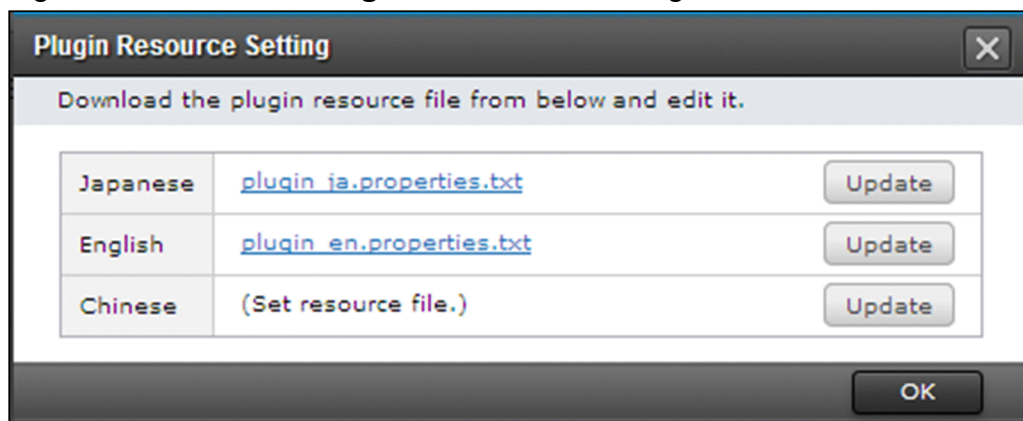
You can set the information displayed for a plug-in by entering settings in a plug-in resource file. You can edit the plug-in resource file directly by downloading the file and overwriting it.

You cannot set display information for plug-ins in released service templates.

#### To set a plug-in resource file:

1. In the **Service Builder Home** window, from the **Custom Plug-in Actions** pull-down menu, select **Set Resources**.
2. In the **Custom Plug-in List** dialog box, select a plug-in, and then click the **Set Resources** button.
3. In the **Set the Plug-in Resources** dialog box, click the link to the plug-in resource file, and download the file.

Figure 6-15: **Set the Plug-in Resources** dialog box



4. Edit the plug-in resource file you downloaded.
5. Click the **Refresh** button, select the plug-in resource file you edited, and upload the file.  
If the plug-in resource file you uploaded is not named `plugin_<language-code>.properties.txt`, an error occurs.
6. In the conformation dialog box, click the **OK** button.



#### Important

When you upload the plug-in resource file, the existing file is overwritten with the contents of the new file. Take care not to upload the wrong file.

*language-code* is a two-character language code (ja, en, or zh) as defined in ISO-639

### Operation result

Display information of the plug-in is set according to the contents of the resource file.

---

## Related topics

- [6.5.2 Format of plug-in resource files](#)
  - [6.5.3 Correspondence between properties in plug-in resource files and information displayed for plug-ins](#)
- 

## 6.5.2 Format of plug-in resource files

A plug-in resource file defines the items displayed in the JP1/AO user interface. The file has the following format:

- The file name of the plug-in resource file is `plugin_language-code.properties.txt`.  
*language-code* is a two-character language code (ja, en, or zh) as defined in ISO-639.
- Define the file contents in the format *property-key delimiting-character setting-value*. As the delimiting character, you can use an equals sign (=), a colon (:), a tab character (\t), or a single-byte space.  
For example, enter a definition in the format `plugin.displayName=TestPlugin`.
- Enter one property key and setting per line.
- Property keys can be 1 to 128 characters long, and can contain the following characters:
  - Single-byte alphanumeric characters
  - Single-byte hyphens (-)
  - Single-byte underscores (\_)
  - Single-byte periods (.)
- Characters must be encoded in UTF-8.
- If you define the same property key in the file more than once, the value of the last occurrence of the property key applies.
- Lines that begin with a hash mark (#) are handled as comments.
- Property keys are case sensitive.
- To specify a character string that contains a forward slash (/), specify two forward slashes (\\) instead.
- Lines that consist only of single-byte spaces are ignored.
- On each line of the plug-in resource file, the property key is the character string from the first character that is not a single-byte space to the character immediately preceding the first delimiting character.
- The setting value is the string from the first non-delimiting character after the delimiting character following the property key to the last character of the line.  
For example, the following line represents the property key `abc` with the setting value `=\tc`:  
`abc\t=\tc`  
However, if the character immediately following the first delimiting character is = or :, the setting value is the character string from the next character that is not a single-byte space or \t to the end of the line.  
For example, the following line in the plug-in resource file represents the property key `abc` with the setting value `=\tc`:  
`abc\t=\t=\tc`
- Surrogate pair characters are ignored.

## 6.5.3 Correspondence between properties in plug-in resource files and information displayed for plug-ins

You can set the display information for plug-ins from the user interface. This information is also defined in the plug-in resource file. The following table lists the correspondence between the display information of a plug-in and the properties in the plug-in resource file.

Table 6-16: Correspondence between properties in plug-in resource file and display information

Plug-in display information	Property in plug-in resource file
Plug-in name	plugin.displayName
Vendor name	plugin.vendorDisplayName
Description	plugin.shortDescription
Input property name / Output property name	property. <i>property-key</i> .displayName
Input property description / Output property description	property. <i>property-key</i> .description

## 6.5.4 Plug-in resource files automatically generated when plug-ins are created

When a plug-in is created, a plug-in resource file is automatically generated. Two types of plug-in resource files can be generated: plug-in resource files that use the same language as the Web browser locale, and plug-in resource files for English. However, if the locale of the Web browser is English, only the one for English is generated.

The values shown in the following table are set for the generated plug-in resource files.

Table 6-17: Default values of display information defined in plug-in resource files (when plug-ins are created)

Display information to be defined	Value set by default	
	Same language as the Web browser locale	Resource file for English that is automatically generated <sup>#</sup>
Vendor name	Value specified when the plug-in is created, copied, or edited	Vendor ID
Plug-in name		Plug-in ID
Description for the plug-in		Blank
Plug-in input property name, plug-in output property name		Property key
Plug-in input property description, plug-in output property description		Blank

#

If the Web browser locale is English, see the *Same language as the Web browser locale* column for the contents of the generated plug-in resource file.

The following gives examples of plug-in resource files that are automatically generated when you create plug-ins.

## Values specified in the window

```
Vendor ID: test.vendor  
Plug-in ID: test.plugin  
Plug-in version: 10.00.00  
Plug-in name: test.plugin  
Vendor name: test.vendor  
Description: This plugin is for testing purposes.
```

## Generated plug-in resource file

```
plugin.vendorDisplayName=test.vendor  
plugin.displayName=test.plugin  
plugin.shortDescription=This plugin is for testing purposes.
```

### 6.5.5 Plug-in resource files updated when plug-ins are edited

If you edit and save a plug-in, the plug-in resource file for the same language as the Web browser locale is updated.

However, if you add or delete the definition of display information or update property keys, the change is also applied (for consistency) to plug-in resource files for languages other than the Web browser locale.

If you add the definition of display information, the values described in the table below are added to the definition in the plug-in resource files for languages other than the Web browser locale. If you update a property key when you update the definition of display information, the values in the plug-in resource files for languages other than the Web browser locale are automatically overwritten and set with the values in the table below. Therefore, if you want to view the old information (the information prior to being overwritten), back up the plug-in resource files for languages other than the Web browser locale.

Table 6-18: Values of display information to set in plug-in resource files

Display information to define	Value to be set
Plug-in input property name, plug-in output property name	Property key
Plug-in input property description, plug-in output property description	Blank

If you delete the definition of display information, the corresponding definition information is also deleted from the plug-in resource files for languages other than the Web browser locale.

To set display information for languages other than the Web browser locale, you need to manually create, edit, and upload the plug-in resource file. When you create a plug-in resource file manually, we recommend that you download and use the plug-in resource file for a locale for which display information has been set.

### 6.5.6 Displaying plug-ins by using a Web browser with a locale for which the plug-in resource file has not been created

If you display plug-ins by using a Web browser with a locale for which the plug-in resource file has not been created, the plug-in resource file for English is loaded, and the window is displayed.

# 7

## Managing plug-ins

In addition to creating or editing plug-ins, you can also copy or delete them.

## 7.1 Copying plug-ins

This section describes how to copy development plug-ins or release plug-ins.

### 7.1.1 Procedure for copying plug-ins

You can copy a development plug-in or release plug-in to create a new development plug-in that retains the settings of the original. Use this procedure when you want to develop a new plug-in based on an existing plug-in, or to create a modified version of an existing plug-in.

#### To copy a plug-in:

1. In the **Service Builder Home** window, from the **Custom Plug-in Actions** pull-down menu, select **Copy**.
2. In the **Custom Plug-in List** dialog box, select the plug-in you want to copy in the **Release** tab or **Develops** tab, and then click the **Copy** button.
3. In the **Copy Custom Plug-in** dialog box, change at least one from plug-in ID, plug-in version, and vendor ID, and then click the **Save** button.

You cannot specify a vendor ID that begins with `com.hitachi.software.dna` because such IDs are reserved by JP1/AO. If the vendor ID of the original plug-in begins with `com.hitachi.software.dna`, the vendor ID and vendor name are removed when the plug-in is copied.

Figure 7-1: **Copy Custom Plug-in** dialog box

**Copy Custom Plug-in**

The vendor ID and vendor name were deleted because the Component on the copy source is using a reserved vendor ID.

Specify custom plug-in attributes. The remote command setting information is required. Drag and drop one or more rows to change the alignment sequence of the input and output properties.

**General** | Property | Remote Command

Key Name: \* dbAddOracleUser\_Win

Version: \* 02.00.00

Vendor ID: \* Enter Vendor ID

Display Name: \* Add an Oracle DB user (Windows)

Vendor Name: Enter Vendor Name

Description: On the execution target server in a Windows environment, start SQL\*Plus and execute the CREATE USER statement to add a d

Tags: Control Database x Oracle Database x

Icon: Restore Default Icon Change

\* Required Save Cancel



#### Tip

To copy a plug-in while editing a service template, select the plug-in you want to copy on the **Developing Plug-in** or **Released Plug-in** tab of the **Flow** tab of the **Service Builder Edit** window, and then click the **Copy** button.

## Operation result

The plug-in is copied and you can then set the definition information for the plug-in.

---

### Related topics

- [6.2.3 Items to set in plug-in definition information](#)
  - [6.2.2 Procedure for editing plug-in definition information](#)
-



## 7.2 Deleting plug-ins

This section describes how to delete development plug-ins or release plug-ins.

### 7.2.1 Procedure for deleting plug-ins

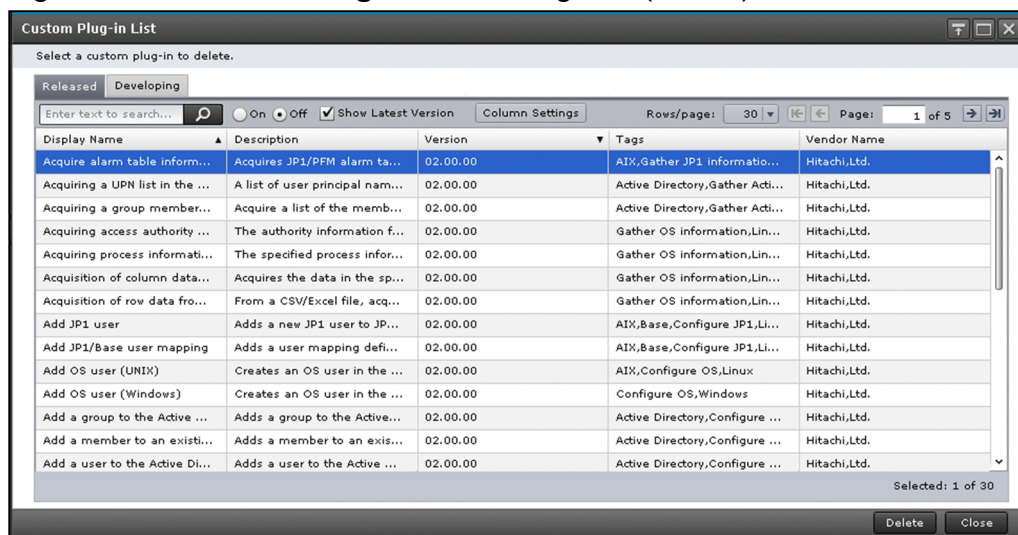
You can delete plug-ins that appear in the **Developing** tab and **Releases** tab in the **Custom Plug-in List** dialog box. When you delete a plug-in, the definition of the plug-in is deleted from the JP1/AO server, and the plug-in disappears from the **Component** area in the **Flow** tab of the **Service Builder Edit** window.

However, you cannot delete a plug-in that is being used by a development service template or a release service template. Nor can you delete a plug-in if a development service template has been built that uses that plug-in as a step. In this scenario, delete the step from the development service template, and build the service template again. You will then be able to delete the plug-in.

#### To delete a plug-in:

1. In the **Service Builder Home** window, from the **Custom Plug-in Actions** pull-down menu, select **Delete**.
2. In the **Release** tab or **Developing** tab of the **Custom Plug-in List** dialog box, select the plug-in you want to delete, and then click the **Delete** button.

Figure 7-2: **Custom Plug-in List** dialog box (delete)



#### Tip

You can delete a plug-in while editing a service template by selecting the plug-in you want to delete in the **Developing Plug-in** or **Released Plug-in** tab of the **Flow** tab of the **Service Builder Edit** window, and then clicking the **Delete** button.

### Operation result

The plug-in is deleted.

# 8

## Validating Service Templates

After creating a service template, you can perform a validation process to make sure that the service template operates as intended in the active environment.

## 8.1 Overview of service template validation

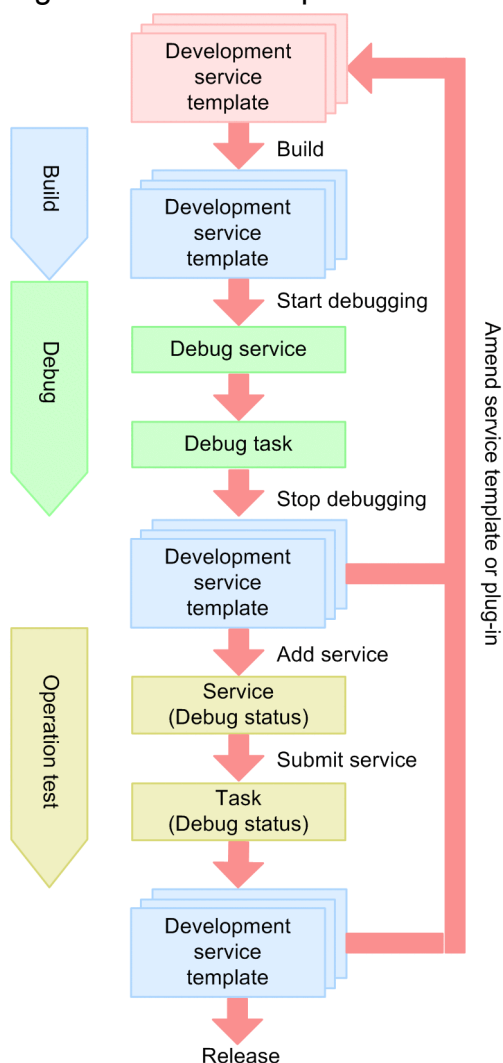
This section describes the general procedure for validating service templates and provides an overview of building, debugging, and operation testing.

### 8.1.1 General procedure for validating service templates

After creating or editing a service template, build and debug the service template to check for problems in flow transitions and plug-in processing. If problems are found, edit the affected service template or plug-in, and then build and debug the service template again. When all problems have been resolved, perform an operation test by executing the service in the development environment.

The following figure shows the general procedure for validating service templates.

Figure 8-1: General procedure for validating service templates



### Building

1. After creating or editing a service template, build the service template to prepare it for the operation check.

## Debugging

1. If the build process is successful, debug the service template to find problems in flow or in the plug-ins. When you debug the service template, a debug service and debug task are generated.
2. Check the execution results of the debug task. If any problem is found, amend the service template, and then rebuild and debug the amended service template.  
Repeat the process of amending, building, and debugging the service template until all problems are resolved.

## Operation test

1. After all problems have been resolved, perform an operation test by creating and executing the service in the development environment.  
Check the execution results. If any problems are found, amend the affected service template or plug-in, and then rebuild and debug the amended service template.

If there are no further problems, release the service template.

Note that you can perform debugging and operation tests on an as-needed basis.

## Related topics for building service templates

- [8.1.2 Overview of building](#)
- [8.2 Building service templates](#)

## Related topics for debugging service templates

- [8.1.3 Overview of debugging](#)
- [8.3 Debugging service templates](#)
- [8.4 Managing debug tasks](#)

## Related topics for testing the operation of service templates

- [8.1.4 Overview of operation tests](#)
- [8.5 Testing the operation of service templates](#)

## 8.1.2 Overview of building

Building is the process of preparing a service template you created or edited in the **Service Builder** window for validation. When building is successful, the service template is packaged so that it can be debugged and used for creating a service.

### Objective

Building is performed to prepare a development service template for validation. The service template you have built is packaged as a debug configuration service template, and then imported to the JP1/AO server.

### Number of executions

You can build a service template any number of times. If the debugging or operation test reveals a problem with the service template, you must repeat the series of operations, from amending the development service template or development plug-in to building, debugging, and testing its operation, until all problems are resolved.

If you edit and rebuild a service template that has already been debugged, JP1/AO deletes the debug service and debug task that were previously generated. If you rebuild a service template after testing its operation, JP1/AO deletes the service that was previously generated and archives the task. For details about the services and tasks archived and deleted during build operations, see [A.1 \(5\) Deletion and archiving of service templates, services, and tasks during build and release operations](#).

## Assigned status

After building, a service template is set to Debug status. Only users in the Admin or Develop role can view and work with service templates in Debug status and the services and tasks created from those service templates.

## Output destinations of service templates

When you build a service template, a service template package is created in one of the following folders with the name *vendor-ID\_name\_version\_d.st*:

In a Windows non-cluster system

*JP1/AO-installation-folder\develop\output*

In a Linux non-cluster system

*/var/opt/jp1ao/develop/output*

In a Windows cluster system

*shared-folder-name\develop\output*

In a Linux cluster system

*shared-folder-name/develop/output*

---

## Related topics

- [2.1 Overview of development service templates and release service templates](#)
  - [8.1.1 General procedure for validating service templates](#)
  - [8.2.1 Procedure for building service templates](#)
  - [A.1 Reference information for build and release operations](#)
- 

## 8.1.3 Overview of debugging

Debugging is the process of using the **Service Builder Debug** window to check the operation of a service template you have built, and identify problems in its flow or plug-ins. When you debug a service template, a debug service and debug task are created. The debug process involves executing this debug task.

If debugging reveals a problem, stop debugging and then edit the affected service template or plug-in and then correct the problem.

## Objective

Debugging is performed to make sure the flow and plug-ins of a service template are working as intended. For example, you can confirm that property mapping is configured correctly and that the flow branches as intended in the design based on the conditions for executing subsequent steps. In the **Service Builder Debug** window, you can:

- Execute debug tasks while checking the flow transitions at all hierarchical levels (including hierarchy flows and repeated execution flows) and the results of plug-in processing.

- Execute debug tasks while making sure that property values are assigned correctly according to the property mapping settings.
- If you detect a problem with a plug-in, you can change property values and return value of any step and re-execute the debug task. This allows you to check the plug-in processing and flow transitions when a given value is assigned as the step-property value or return value.

## Number of executions

You can debug a service template any number of times. If debugging reveals a problem with the service template, you must repeat the series of operations, from amending the development service template or development plug-in to building and debugging, until all problems are resolved.

## Debug services

A debug service is generated and executed during debugging of a service template. One debug service is generated per service template. If you debug a service template that has already been debugged, JP1/AO deletes the previously generated debug service, and then creates a new one.

Note that debug services appear in the Service Name column in the **Tasks** window (**Debug** tab), but do not appear in the **Services** window.

## Debug tasks

A debug task is generated for a debug service during debugging of a service template. If you debug a service template that has already been debugged, JP1/AO deletes the previously generated debug task, and then creates a new one.

Debug tasks appear in the **Service Builder Debug** window and **Tasks** window (**Debug** tab). Only users in the Admin or Develop role can view and work with debug tasks.

Note that debug tasks do not appear in the task summary.

---

## Related topics

- [2.1 Overview of development service templates and release service templates](#)
  - [8.1.1 General procedure for validating service templates](#)
  - [8.3.2 General procedure for debugging service templates](#)
  - [8.3.3 Functions used during debug operations](#)
  - [8.3.4 Example of debugging service templates](#)
- 

## 8.1.4 Overview of operation tests

An operation test is the process of creating a service from a service template you have built, and then executing the service in the development environment to confirm that the service template is ready for real-world use.

If the operation test reveals a problem, you can edit the service template or plug-in in the **Service Builder** window.

## Objective

An operation test is performed by executing a service generated from the service template to confirm that the service operates correctly in the active environment. You can also test the usability of the service template by creating and executing services from the **Services** window in a way that reflects real-world use. For example, you can specify a

schedule type of the service to confirm that the operation is appropriate for your purpose, and to make sure that the properties are visible or values can be edited.

## Number of executions

You can perform any number of operation tests for a given service template. If the operation test reveals a problem with the service template, you must repeat the series of operations, from amending the development service template or development plug-in to building, debugging, and testing its operation, until all problems are resolved.

---

### Related topics

- [2.1 Overview of development service templates and release service templates](#)
  - [8.1.1 General procedure for validating service templates](#)
  - [8.5.1 Procedure for testing the operation of service templates](#)
  - Managing services in the JP1/Automatic Operation Administration Guide
  - Executing services in the JP1/Automatic Operation Administration Guide
  - Managing Tasks in the JP1/Automatic Operation Administration Guide
-

## 8.2 Building service templates

This section describes how to build service templates.

### 8.2.1 Procedure for building service templates

When you select and build a service template, a service template package is created and imported to the JP1/AO server. You can then debug and test the service template.

#### ! Important

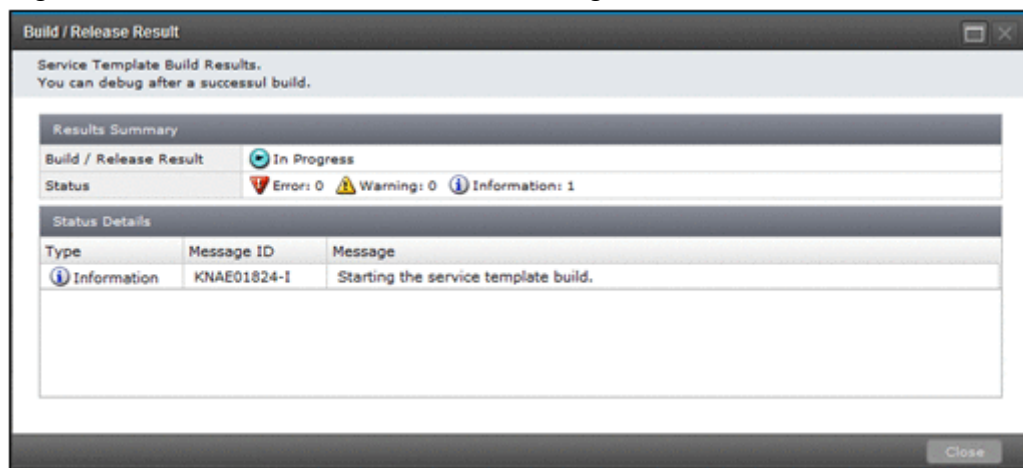
If you edit the definition of a plug-in that is placed as a step in a flow, the mapping settings defined for the step might become inconsistent with the step properties. In this case, an error will occur when you build the service template. If an error occurs, resolve the mismatch by reviewing the property settings and plug-in placement.

#### To build a service template:

1. In the **Service Builder Edit** window, click **Debug**.
2. In the confirmation window, click **OK**.

The results of the build process appear in the **Build / Release Result** dialog box. If the build is successful, the **Perform Debugging** dialog box appears.

Figure 8-2: **Build / Release Result** dialog box



3. If you want to then debug the service template, click **OK** in the **Perform Debugging** dialog box. For details about how to start debugging, see [8.3.5 Procedure for starting debugging](#).

If you do not want to debug the service template, click **Cancel**.

If an error occurs during the build process, a message dialog box indicating the cause of the error prompts you to amend the flow. At this time, the **Error** button appears at the upper right of the **Service Builder Edit** window. After closing the message dialog box, you can view the message by clicking the **Error** button. This button remains on screen until you close the **Service Builder Edit** window.

If the service template you are working with has already been released or deleted by another user, an error occurs and the build process fails.



---

## Related topics

- [1.2.4 Notes when an interrupt operation is performed in the \*\*Service Builder\*\* window](#)
  - [8.1.1 General procedure for validating service templates](#)
  - [8.1.2 Overview of building](#)
  - [A.1 Reference information for build and release operations](#)
-

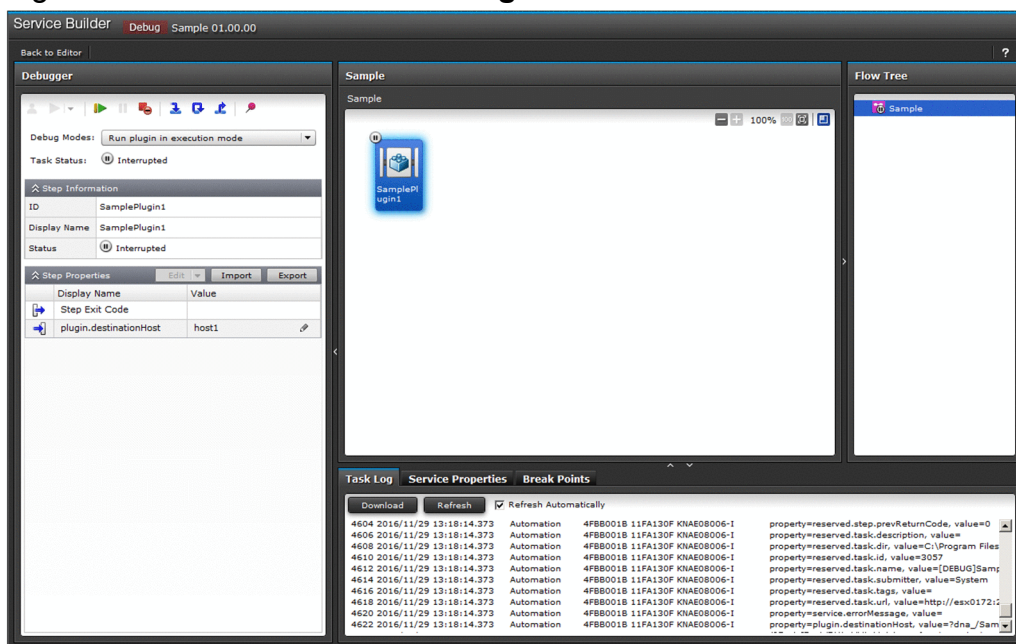
## 8.3 Debugging service templates

You can use the **Service Builder Debug** window to debug service templates.

### 8.3.1 Service Builder Debug window

The **Service Builder Debug** window is used to debug service templates. This window is displayed by clicking **OK** in the **Perform Debugging** dialog box.

Figure 8-3: **Service Builder Debug** window



The following describes the items displayed in this window.

#### Back to Editor button

When you click this button, the **Service Builder Debug** window closes and the **Service Builder Edit** window appears. At this time, the debug task that is running is forcibly stopped.

#### ! Important

You must use the **Back to Editor** button to close the **Service Builder Debug** window. If you use another button to close the Web browser, the debug task will remain running and you will be unable to edit the service template on which it is based. If you used a button other than the **Close** button to close the **Service Builder Debug** window, use the **Tasks** window to stop the debug task.

#### Debug area

This area is used to specify settings when executing a debug task and performing operations on a debug task that is in progress.

#### Enter Response button

You can enter a response to a debug task that is waiting for a response.

### **Resubmit** pull-down menu

You can retry a debug task that is in Completed or Failed status. You can select the retry method from **Retry the Task**, **Retry From Failed Step**, or **Retry From Next To Failed Step**.

### **Resume** button

Resumes execution of an interrupted debug task.

### **Interrupt** button

Interrupts execution of a debug task.

### **Forcibly Stop** button

Forcibly stops execution of a debug task.

### **Step Into** button

Executes the interrupted step up to the next point in that step at which execution can be interrupted.

### **Step Over** button

Executes the interrupted step and then stops execution at the point in the next step at which step execution is interrupted.

### **Step Return** button

Executes the steps in the flow at the second or lower level and then stops execution at the first point in the upper flow at which step execution can be interrupted.

### **Set Break Point** button

Sets breakpoints for a selected step. If breakpoints have already been set for the step you selected, you can cancel them.


### **Debug Modes** pull-down menu

You can specify whether to perform plug-in processing when executing a step.

### **Step Information** area

Displays the step ID, step name, and status of the selected step.

### **Step Properties** area

Displays the property values of the selected step. You can edit the property values by clicking . For input properties of a step that uses service components, click the **Edit** pull-down menu to assign values in the **Create Service** or **Submit Service** window. Click **Import** or **Export** to import or export the step properties by using a property file.

### Flow area

Displays the execution order of the steps as a flow. You can check the step status, whether breakpoints are set, and whether the step can be interrupted.

### **Flow Tree** area

This area is used to select the level of flow to be displayed in the flow area. The flow names defined in the service template are displayed in hierarchical layers. If you click a flow name, the flows under that flow appear in the flow area.

### **Task Log** tab

Displays the task log data for the debug task that is in progress. You can download the contents of the task log.

### **Service Properties** tab

Displays a list of service properties specified in the service template for each property group. However, variables are not displayed.


## Break Points tab

Displays the names of steps for which breakpoints are set, and the level of the flow where the step is placed. Click the **Remove All Break Points** button to cancel all the breakpoints. Right-click a displayed step to cancel breakpoints or select that step in the flow area.

## 8.3.2 General procedure for debugging service templates

If the build process is successful, debug the service template to find problems in its flow or plug-ins.

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

1. If you do not expect any problems when executing all plug-ins in the service template, you must first execute the debug task without pausing between steps.  
Make sure that there are no problems with the flow transitions or the processing of the plug-ins.  
If the service template contains any component that you do not want to execute at this time, skip this step and start from step 2.
2. If you find a problem with a flow transition or the processing of a plug-in, interrupt the execution between steps during debugging to identify the precise location and nature of the problem. You can set breakpoints for any step. If necessary, you can also check the operation by changing the values of input and output properties. You can debug the service template any number of times from the **Resubmit** pull-down menu  in the **Service Builder Debug** window.
3. In the **Service Builder Edit** window, amend the service template.
4. Build and debug the service template again, repeating steps 1 to 4 until all problems are resolved.
5. Make sure that all problems have been resolved, and then finish the debugging.



### Tip

After executing a debug task, you do not need to perform the build operation again if you repeat the debugging without amending the service template.

## Related topics

- [8.1.1 General procedure for validating service templates](#)
- [8.1.2 Overview of building](#)
- [8.1.3 Overview of debugging](#)
- [8.3.3 Functions used during debug operations](#)
- [8.3.4 Example of debugging service templates](#)

## 8.3.3 Functions used during debug operations

The table below describes the functions relating to debugging of service templates. When debugging service templates, use the functions that are appropriate for your purpose.

Table 8-1: Functions used during debug operations

Function	Description	Reference
Setting up debug tasks	You can specify the definition information of the debug service and debug task, the task log output level, and information about debug service properties.	<a href="#">8.3.5 Procedure for starting debugging</a>
Setting up step execution	<p>You can execute a debug task without pausing between steps (like execution of normal tasks), or by interrupting a step before and after processing of the plug-in. A combination of the following methods can be used for steps in a debug task.</p> <ul style="list-style-type: none"> <li>Do not stop after each step Like execution of a normal task, processing of the debug task proceeds without pausing between steps. You can check the execution results of the debug task and identify steps where problems occur.</li> <li>Stop after each step Execution of the debug task pauses before or after plug-in processing for a selected step. This allows you to check the property values and return values.</li> </ul>	<a href="#">8.3.9 Operations for interrupting step executions during debugging</a>
Skipping plug-in processing	You can skip plug-in processing and continue processing as if execution of the step has been completed.	<a href="#">8.3.11 Procedure for skipping plug-in processing during debugging</a>
Checking property mapping settings	You can display the values of step properties in the <b>Debug</b> area. By viewing this information together with the service property values displayed on the <b>Service Properties</b> tab, you can check whether property mapping is configured correctly.	<a href="#">8.3.13 Procedure for checking property mapping settings during debugging</a>
Changing information in a step	<p>You can change a property value and return value of a step to any value.</p> <ul style="list-style-type: none"> <li>You can interrupt a step before the plug-in processing for that step is performed, and then change the values of input properties.</li> <li>After plug-in processing is skipped or executed, you can interrupt a step and then change the values of output properties or a return value.</li> </ul>	<a href="#">8.3.14 Procedure for changing step property values or return value during debugging</a>
Retrying tasks	<p>You can use the following ways to retry a failed debug task:</p> <ul style="list-style-type: none"> <li>Resubmit You can execute the debug task as a new one without rebuilding it.</li> <li>Retry From Failed Step You can resume task execution from the failed step.</li> <li>Retry From Next To Failed Step You can resume task execution from the next step, as if the failed step had finished normally.</li> </ul>	<a href="#">8.3.19 Procedure for debugging a service template again without rebuilding</a> , <a href="#">8.3.20 Procedure for retrying a task from a failed step during debugging</a> , and <a href="#">8.3.21 Procedure for retrying a task from the step after the failed step during debugging</a>
Displaying flow information	<p>You can use the following ways to display the flow of a debug task:</p> <ul style="list-style-type: none"> <li>Flow view In the <b>Service Builder Debug</b> window and <b>Tasks</b> window, you can view the status of steps and the flow transitions at each hierarchical level.</li> <li>Flow Tree view In the <b>Service Builder Debug</b> window, you can view flow hierarchies in tree format. You can also identify interrupted steps in each flow hierarchy.</li> </ul>	<a href="#">8.3.22 Displaying debug task flow</a> , <a href="#">8.3.23 Displaying the flow tree of a debug task</a> , <a href="#">8.4 Managing debug tasks</a>
Displaying task logs	The contents of the task log can always be displayed on the <b>Task Log</b> in the <b>Service Builder Debug</b> window. You can configure JP1/AO to automatically refresh and download the contents of the task log.	<a href="#">8.4.3 Procedure for checking task log entries for debug tasks</a>
Managing debug tasks	<p>You can do the following for debug tasks:</p> <ul style="list-style-type: none"> <li>Check the progress</li> <li>Check detailed information</li> <li>Stop execution</li> <li>Forcibly stop</li> <li>Delete</li> </ul>	<a href="#">8.4 Managing debug tasks</a>

---

## Related topics

- [8.3.2 General procedure for debugging service templates](#)
  - [8.3.4 Example of debugging service templates](#)
- 

### 8.3.4 Example of debugging service templates

When debugging a service template, users need to adjust various settings according to the aspects of the service template they want to check.

The following shows an example of the operations a user performs when debugging a service template.

This example describes the procedure for debugging a service template in the following scenario:

#### Problems with service template being debugged

Problem 1:

There is a problem with mapping settings between a service property and input property of Step A.

#### User objectives

Objective 1:

Check the execution results of other steps before amending the service template.

Objective 2:

If a problem is found in the property mapping, prevent processing of the plug-in in the step.

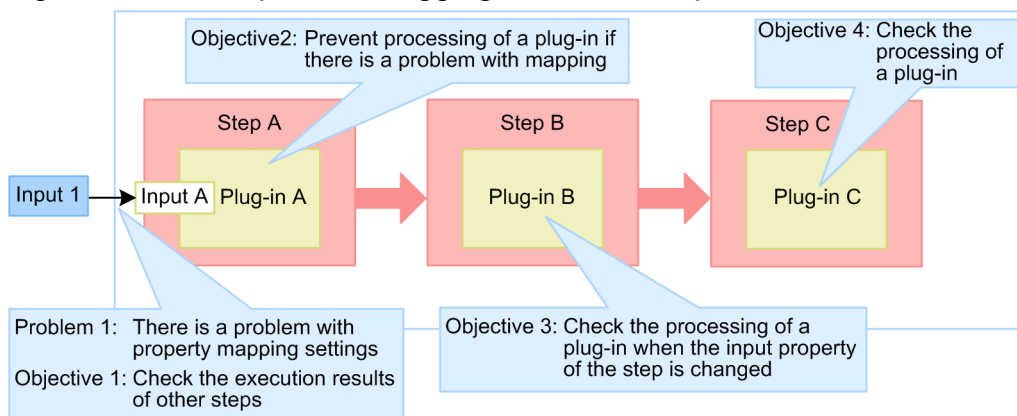
Objective 3:

Change the step property of Step B, and then check the results of processing of Plug-in B.



Objective 4:

Check the results of processing of Plug-in C.

Figure 8-4: Example of debugging a service template



1. The debug task starts, and then Step A is automatically interrupted before processing of Plug-in A is performed. When Step A is interrupted, in the **Debug** area, check the value of step property *Input A*. Make sure that the value of service property *Input 1* has been mapped correctly in the **Service Properties** tab.  
This process allows you to identify the problem with the mapping settings of step property *Input A* (Problem 1).

2. A problem is found in the property mapping. Therefore, according to Objectives 1 and 2, you must skip the processing of Plug-in A to proceed with the debug task. In the **Debug** area, from the **Debug Modes** pull-down menu, select **Run plugin in dry-run mode**.
3. To change the step property value and perform processing of Plug-in B as described in Objective 3, click the **Step Over** button (  ) to advance the debug task up to the point before the processing of Plug-in B.  
Processing of Plug-in A is skipped and execution of Step A ends. Then, Step B is interrupted before the processing of Plug-in B is performed.
4. In the **Debug** area, from the **Debug Modes** pull-down menu, select **Run plugin in execution mode**.
5. Change the value of the input property of Step B.
6. In the **Debug** area, click the **Resume** button (  ).  
After processing of Plug-in B is performed, Step C is executed according to the subsequent-step execution condition, and then the debug task terminates.
7. To meet Objectives 3 and 4, check the execution results of Steps B and C in the **Debug** area, flow area, and **Task Log** tab.
8. In the **Service Builder Debug** window, click **Back to Editor**.  
The **Service Builder Debug** window closes.
9. In the **Flow** tab of the **Service Builder Edit** window, amend the property mapping settings for Step A.
10. Save the service template, and then debug the service template again.

---

## Related topics

- [8.1.1 General procedure for validating service templates](#)
  - [8.1.2 Overview of building](#)
  - [8.1.3 Overview of debugging](#)
  - [8.3.2 General procedure for debugging service templates](#)
  - [8.3.3 Functions used during debug operations](#)
  - [8.3.9 Operations for interrupting step executions during debugging](#)
- 

## 8.3.5 Procedure for starting debugging

If the build process of a service template is successful, you can start debugging. To do this, you must specify the definition information for the debug service and debug task, task log output level, and debug service property information.

Note that schedule type is set to immediate execution when debugging is performed.

### To start debugging:

1. In the **Perform Debugging** dialog box, specify the definition information for the debug service and debug task, task log output level, and debug service property information.  
To specify properties, from the **Edit** pull-down menu, select **From Create Service Window** or **From Create Request Window**, and then specify the properties in the window that appears. You can also read properties from the property file.



Figure 8-5: **Perform Debugging** dialog box

Service Name: \* [DEBUG]Add JP1/Base monitoring settings

Tags: Base x Configure JP1 x

Task Name: \* [DEBUG]Add JP1/Base monitoring settings\_20151030111522

Task Description: Enter task description.

Service Group: DefaultServiceGroup

Task Log Level: Debug

**Properties** Edit Restore Default Import Export

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

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

Display Name	Key Name	Value	Description	Scope
Monitor server information				
List of host names fo...	common.targetHostList		Specify the host n...	local
JP1/Base installation ...	jp1base.targetBaseP...	C:\Program Files\Hit...	Specify the install...	local
Path to JP1/Base (Wi...	jp1base.targetForwar...	C:\Program Files\Hit...	Specify the locati...	local
Path to JP1/Base (Wi...	jp1base.targetConfP...	C:\Program Files\Hit...	Specify the locati...	local
Path of forwarding fil...	jp1base.targetForwar...	/etc/opt/jp1base/con...	Specify the full pa...	local
Path of configuration ...	jp1base.targetConfP...	/etc/opt/jp1base/conf	Specify the full pa...	local
The path of JP1/Base	in1base.targetComm...		Specifies as a ful...	local

\* Required OK Cancel

2. Click **OK**.

## Operation results

The **Service Builder Debug** window appears, and then execution of the debug task starts.

After the debug task starts, the first step is interrupted before processing of the plug-in for that step is performed. You can resume execution of the step by clicking **Resume** in the **Debug** area.

### Important

- The **Perform Debugging** dialog box shows the status of the service template when it was built the last time by the current debugging user. If another user edits and builds the same service template after the debugging user, the changes do not apply to the contents of the service template shown in the **Perform Debugging** dialog box.
- A debug task is forcibly terminated in the following circumstances:
  - The JP1/AO server stops.
  - Failover to another node in a cluster system occurs.
- If you close your Web browser during the debugging, the debug task is interrupted, and you will be unable to execute subsequent processing. In this case, you need to log in again and stop the debug task in the **Debug** area of the **Tasks** window. Note that the **Service Builder Debug** window you closed does not re-appear. To execute the debug task again, you need to rebuild the service template, and then restart debugging.
- After you build the service template, if another user builds the same service template before you start the debugging, an error occurs when you click **OK** in the **Perform Debugging** dialog box.
- A single JP1/AO system can execute a maximum of 10 plug-ins concurrently in a debug task. For details about the maximum number of plug-ins that can be executed concurrently and what happens when the



number exceeds this limit, see *Maximum number of concurrently executable plug-ins in a task* in the *JP1/Automatic Operation Administration Guide*.

- The status of debug tasks is subject to JP1 event reporting and email notification.

## Related topics

- [8.3.6 Settings used when starting debugging](#)
- [8.3.3 Functions used during debug operations](#)
- [8.3.7 Procedure for debugging without pausing between steps](#)
- [8.3.9 Operations for interrupting step executions during debugging](#)

## 8.3.6 Settings used when starting debugging

In the **Perform Debugging** dialog box, you can specify the definition information for the debug service and debug task, task log output level, and debug service property information.

Table 8-2: Items displayed in the **Perform Debugging** dialog box

Item	Description
Service Name	Enter the name of the debug service. The default is [DEBUG] <i>name-of-service-template-to-be-executed</i> .
Tags	Enter the tag of the debug service. The default is <i>tag-of-service-template-to-be-executed</i> .
Task	Enter the name of the debug task. The default is [DEBUG] <i>name-of-service-template-to-be-executed_current-time(YYYYMMDDhhmmss)</i> .
Description	Enter the description of the debug task. This field is empty by default.
Service Group	Select the service group in which to register the debug service. The default is DefaultServiceGroup.
Task Log Level	Select the level of messages output to the task log. The default is 40. The log output level you select applies only to the debug task. It has no effect on the existing, shared, built-in service property (com.hitachi.software.dna.sys.task.log.level).
Properties	The values of the input properties of the service are displayed. Default values are assigned according to the service property definitions set when creating the service template. To change property values, from the <b>Edit</b> pull-down menu, select <b>From Create Service Window</b> or <b>From Create Request Window</b> , and then change the properties in the window that appears. You can also click <b>Import</b> to enter values from the property file or click <b>Export</b> to output the specified values. Note that the property values set in this area are specific to the debug task, and do not affect service share properties elsewhere in the system. Therefore, the values you set will not affect other services that reference the service share properties. To reset the service property values to the default values specified for the service template, click <b>Restore Default</b> .

## Related topics

- Overview of property files in the JP1/Automatic Operation Administration Guide


## 8.3.7 Procedure for debugging without pausing between steps

When you start the debug task, step execution is interrupted before processing of the plug-in contained in the first step is performed. Then, like normal tasks, you can execute the debug task without pausing between steps. At this time, you cannot change step property values or the return value.

### Procedure for debugging without pausing between steps:

1. If breakpoints are set for the step, on the **Break Points** tab, click the **Remove All Break Points** button.
2. In the **Debug** area, from the **Debug Modes** pull-down menu, select the item appropriate for what you want to do with the plug-ins.

For example, if you want to confirm that the task terminates normally or check for problems with flow transitions, select **Run plugin in execution mode**. If you want to skip processing of all plug-ins up to the step with a breakpoint specified, select **Run plugin in dry-run mode**.

3. In the **Debug** area, click the **Resume** button (  ).

### Operation results

The debug task is executed without pausing up to the specified breakpoint. In the **Service Builder Debug** window, check the execution results and, if necessary, edit the service template or plug-ins.

---

#### Related topics

- [8.3.5 Procedure for starting debugging](#)
  - [8.3.6 Settings used when starting debugging](#)
  - [8.3.22 Displaying debug task flow](#)
  - [8.4.3 Procedure for checking task log entries for debug tasks](#)
- 

## 8.3.8 Timing with which step execution can be interrupted

A step can be interrupted before and after processing of a plug-in contained in that step is performed.

*Before processing of a plug-in* means the time immediately after input properties are generated after step execution is started. At this time, the status of the step is Interrupted.

*After processing of a plug-in* means the time immediately after output properties and return value are output. At this time, the status of the step is Interrupted (After Execution).

When a step is interrupted, you can change the step property values and return value. This allows you to check the plug-in processing and flow transitions by specifying any values for step properties or return value, thus reviewing problems with plug-in processing and flow. If you interrupt a step before processing of a plug-in, you can then select whether to perform the plug-in processing.

Note, however, that step execution cannot be interrupted for some plug-ins.



#### Tip

- Immediately after debugging starts, step execution is automatically interrupted before processing of the plug-in in the first step.

- If a plug-in returns a value of 65 or greater during processing, debug task processing continues even if you attempt interruption after processing of the plug-in. This might occur if the value of an input property of the step is specified incorrectly or if JP1/AO is unable to connect to the OS of the operation target device.






## Related topics




- [8.3.9 Operations for interrupting step executions during debugging](#)
- [8.3.10 Step information that can be changed when step execution is interrupted](#)

## 8.3.9 Operations for interrupting step executions during debugging

When you execute a debug task, you can specify the next point to go (where step execution was interrupted) in the **Debug** area. The table below describes the operations to execute steps during debugging. Perform appropriate operations to proceed with debugging.

Table 8-3: Operations for interrupting step executions during debugging


Operation	What to do	Available when:	Description	Example use case
Resume <sup>#</sup>	Click the <b>Resume</b> button (  ).	The step is interrupted.	Resumes execution of the interrupted step. Processing starts at the point before processing of a plug-in in the nearest step with a breakpoint set and continues through to the end of the debug task.	<ul style="list-style-type: none"> <li>• You want to execute the debug task up to the step with a breakpoint set.</li> <li>• You want to execute the debug task through to the end without pausing between steps.</li> </ul>
Interrupt	Click the <b>Interrupt</b> button (  ).	The step is in progress.	Executes the step up to the nearest point at which the step can be interrupted, and then interrupts the step.	<ul style="list-style-type: none"> <li>• A debug task is executed by mistake without specifying step interrupts.</li> </ul>
Step into <sup>#</sup>	Click the <b>Step Into</b> button (  ).	The step is interrupted.	Executes the step up to the next point at which the step can be interrupted, and then interrupts the step. For a layering step, repeated step, or service step, the steps in the subordinate flow are also interrupted.	<ul style="list-style-type: none"> <li>• You want to proceed with debugging while checking step information before and after plug-in processing.</li> <li>• You want to proceed with debugging while changing the step's output property values or return value.</li> </ul>
Step over <sup>#</sup>	Click the <b>Step Over</b> button (  ).	The step is interrupted.	Executes the step and then stops execution at the first point in the next step at which execution can be interrupted. For a layering step, repeated step, or service step, the steps in the subordinate flow are executed without interruption.	<ul style="list-style-type: none"> <li>• You want to perform debugging while checking the results of plug-in processing step by step.</li> <li>• You want to perform debugging while changing input properties of the step and checking plug-in processing.</li> </ul>
Step return <sup>#</sup>	Click the <b>Step Return</b> button (  ).	The step is interrupted.	In most cases, this operation is used for a flow subordinate to a layering step, repeated step, or service step. This operation executes steps up to the first	<ul style="list-style-type: none"> <li>• You want to check only the first step in the repeated processing, and skip other steps.</li> </ul>

Operation	What to do	Available when:	Description	Example use case
Step return <sup>#</sup>	Click the <b>Step Return</b> button (  ).	The step is interrupted.	point at which execution can be interrupted in the upper flow and interrupts execution there. For the top-level flow, this operation executes steps to the end of the debug task.	<ul style="list-style-type: none"> <li>You want to check only the first step in the repeated processing, and skip other steps.</li> </ul>
Break points <sup>#</sup>	<ul style="list-style-type: none"> <li>To set breakpoints: Select the step for which you want to set breakpoints, and then click the <b>Set Break Point</b> button (  ).</li> <li>To cancel the breakpoint settings: Select the step for which you want to cancel the breakpoint settings, and then click the <b>Set Break Point</b> button (  ).</li> <li>To cancel all breakpoint settings: On the <b>Break Points</b> tab, click the <b>Remove All Break Points</b> button.</li> </ul>	The <b>Service Builder Debug</b> window is displayed.	Sets breakpoints before and after plug-in processing in the selected step. When you perform the resume operation, step execution continues up to the breakpoint you have set.	<ul style="list-style-type: none"> <li>You want to check step information before and after plug-in processing for a certain step.</li> <li>You want to skip plug-in processing for a certain step.</li> </ul>

#

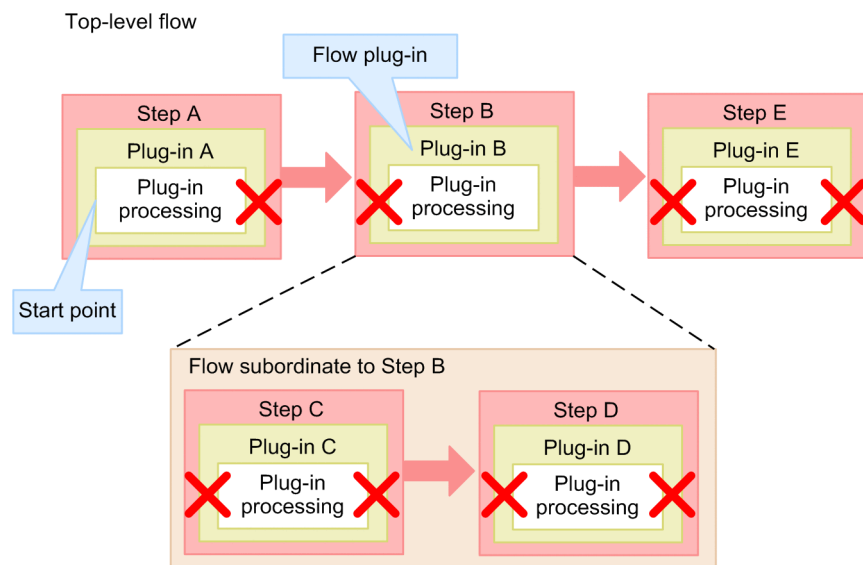
You can perform the same operation by right-clicking the step icon in the flow area.

The figures below display the points at which step execution is interrupted when the step-into, step-over, and step return operations are performed.

Note that if you perform the step-into, step-over, or step return operation,  appears on the icon of the next step in the flow area.

## If step-into operation is repeated after the debug task started

Figure 8-6: Points at which execution is interrupted by step-into operation



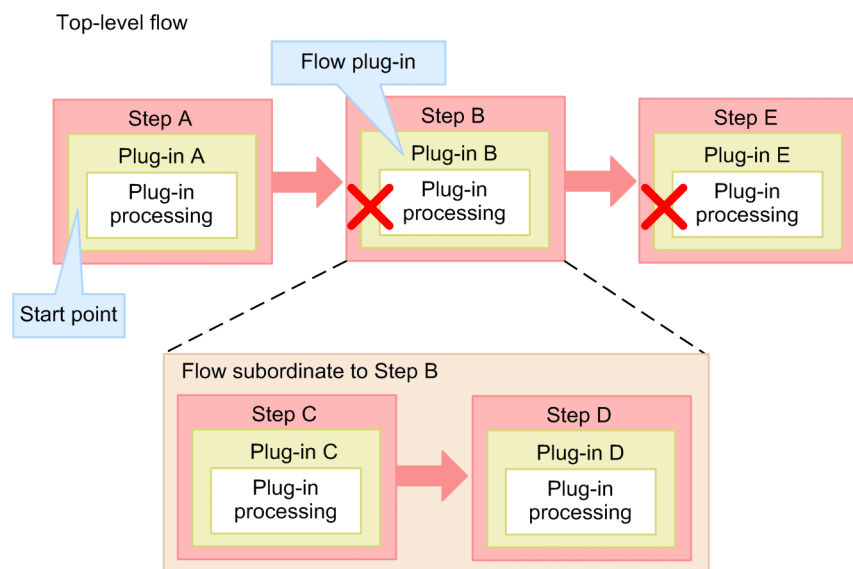
Legend:

**X** : Point at which execution is interrupted when step-in operation is repeated

Step execution is interrupted before and after processing of all plug-ins including those subordinate to the flow plug-in. However, execution cannot be interrupted after the flow plug-in has started. Therefore, execution is interrupted before plug-in processing of Step E, rather than after plug-in processing of Step B.

## If step-over operation is repeated after the debug task has started

Figure 8-7: Points at which execution is interrupted by step-over operation



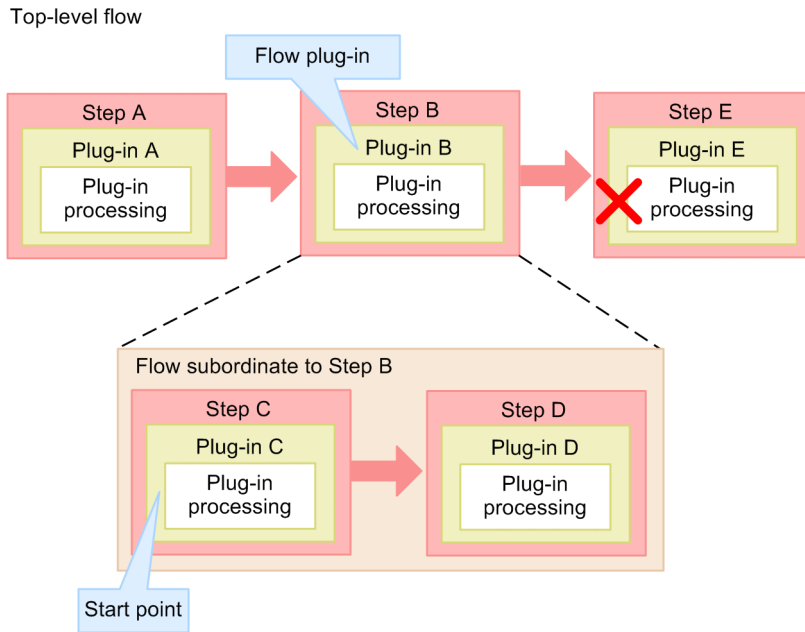
Legend:

**X** : Point at which execution is interrupted when step-over operation is repeated

Execution is interrupted before plug-in processing of each step. However, execution of steps is not interrupted in the flow that is subordinate to the flow plug-in.

## If step-return operation is performed for the flow subordinate to the flow plug-in

Figure 8-8: Points at which execution is interrupted by step-return operation



Legend:

**X** : Point at which execution is interrupted when step-return operation is repeated

After all steps in the flow subordinate to the flow plug-in are executed, processing returns to the upper-level flow, and then execution is interrupted before plug-in processing of Step E.

### Related topics

- [8.3.8 Timing with which step execution can be interrupted](#)
- [8.3.10 Step information that can be changed when step execution is interrupted](#)
- [8.3.11 Procedure for skipping plug-in processing during debugging](#)
- [8.3.12 Plug-ins that cannot be interrupted or skipped during debugging](#)
- [8.3.14 Procedure for changing step property values or return value during debugging](#)

## 8.3.10 Step information that can be changed when step execution is interrupted

You can change step information for the debug task while step execution is interrupted during debugging. If problems with plug-ins are found during debugging, you can check the operation by changing property values and the return value of the step.

Step information you can change varies depending on when the step is interrupted.

Table 8-4: Step information that can be changed while a step is interrupted

Item	If the step is interrupted before plug-in processing		If the step is interrupted after plug-in processing
	If plug-in processing is performed	If plug-in processing is skipped	
Values of input properties of the step <sup>#</sup>	Y	Y	N
Values of output properties of the step <sup>#</sup>	N	N	Y
Return value of the step	N	N	Y

Legend:

Y: Can be changed. N: Cannot be changed

<sup>#</sup>

You cannot change the information in the following cases:

- Property mapping is configured.
- The property of the step contained in the service component is set to be hidden or prohibited from editing in the **Crate Service** window or **Submit Service** window.





## Related topics

- [8.3.8 Timing with which step execution can be interrupted](#)
- [8.3.12 Plug-ins that cannot be interrupted or skipped during debugging](#)
- [8.3.14 Procedure for changing step property values or return value during debugging](#)

## 8.3.11 Procedure for skipping plug-in processing during debugging

You can skip processing of plug-ins that you do not want to execute during debugging. To skip plug-in processing, specify the output property values and return value of the step on the assumption that the plug-in processing is performed. This allows you to assess the effect the subsequent-step execution conditions have on the status transitions of steps and tasks, flow transitions, and the processing of subsequent steps.

### To skip plug-in processing:

1. Interrupt the step that contains the plug-in you want to skip before processing of that plug-in is performed.
2. In the **Debug** area, from the **Debug Modes** pull-down menu, select **Run plugin in dry-run mode**.
3. Execute the step until plug-in processing is performed, and then interrupt execution. If breakpoints are set for the step containing the plug-in you want to skip, click the **Resume** button (  ). If no breakpoints are set, click the **Step Into** button (  ).
4. In the **Step Properties** area, click  for the return value, and then specify the desired value.  
The return value you specify determines the status transitions and the flow transitions of steps and tasks, subject to the subsequent-step execution conditions.
5. If necessary click  for an output property, and then specify the desired value.



## Tip

If you have configured output property mapping, the output property value you specify for the step is applied to the service property (output property or variable) to which the output property is mapped. If you do not specify a value, the service property (output property or variable) to which the output property is mapped will have a null value.

## Operation results

If you perform an operation that executes a process, debug task processing goes on according to the specified return value and output property.

## Related topics

- [8.3.5 Procedure for starting debugging](#)
- [8.3.9 Operations for interrupting step executions during debugging](#)
- [8.3.12 Plug-ins that cannot be interrupted or skipped during debugging](#)
- [8.3.14 Procedure for changing step property values or return value during debugging](#)
- [3.4.4 Overview of subsequent step conditions](#)

## 8.3.12 Plug-ins that cannot be interrupted or skipped during debugging

Some plug-ins cannot be interrupted even if you attempt to interrupt step execution during debugging. If one of these plug-ins is encountered during debugging, the flow automatically advances to the succeeding step.

Table 8-5: Ability to interrupt plug-ins

No.	Plug-in name		Can be interrupted	Can be skipped
1	Basic plug-ins	General command plug-in	Y	Y
2		File-transfer plug-in	Y	Y
3		Repeated Execution Plug-in	Y	Y
4		Email Notification Plug-in	Y	Y
5		User-Response Wait Plug-in	Y	Y
6		Standard Output Plug-in	Y	Y
7		Terminal connect plug-in	Y	Y
8		Terminal command plug-in	Y	Y
9		Terminal disconnect plug-in	Y	Y
10		Flow plug-in	C	N
11		Interval plug-in	Y	Y
12		Branch by returncode plug-in	C	N
13		Test value plug-in	Y	Y
14		Abnormal-end plug-in	Y	Y



No.	Plug-in name		Can be interrupted	Can be skipped
15	Basic plug-ins	Branch by property value plug-in	C	N
16		JavaScript plug-in	Y	Y
17		File Export Plug-in	Y	Y
18		Web Client Plug-in	Y	Y
19	Content plug-ins		Y	Y
20	Service component		C	N <sup>#</sup>
21	Incompatible steps in service templates created in versions of JP1/AO earlier than 10-10		N	N

Legend:

Y: Can be interrupted or skipped. N: Cannot be interrupted or skipped. C: Can be interrupted before the plug-in processing is performed, but cannot be interrupted after the plug-in is executed.

#

A step that contains service components cannot be skipped. However, whether steps in the subordinate flow can be skipped is determined by plug-ins contained in each step.

### 8.3.13 Procedure for checking property mapping settings during debugging

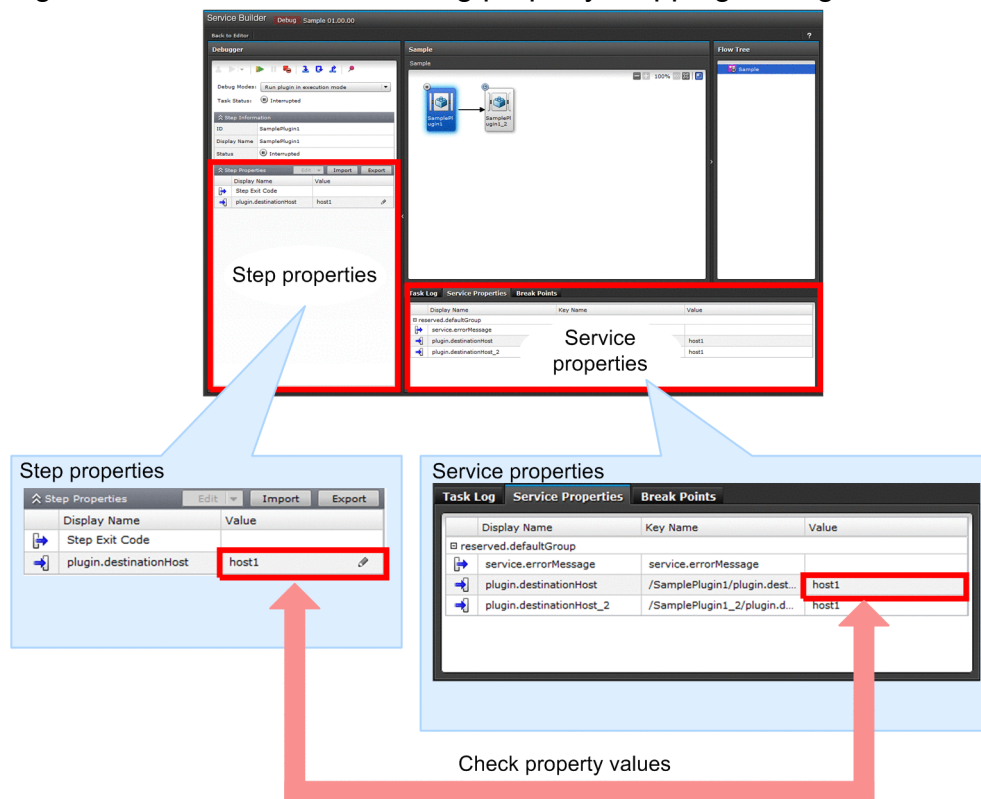
During debugging, you can check whether the values of service properties and step properties are mapped as intended.

By viewing the **Debug** area and **Service Properties** tab, you can make sure that the same values are assigned to step properties and the service properties to which they are mapped.

To check the mapping settings during debugging, check the property values at the following timing:

- For mapping between step input properties and service input properties or variables:  
Check the property values while the step is interrupted before the step processing is performed.
- For mapping between step output properties and service output properties or variables:
  - If step processing is skipped, check the property values when the step is resumed after you specify the output properties and return value of the step.
  - Check the property values when resuming a step that was interrupted after step processing.

Figure 8-9: Window for checking property mapping settings



## To check the property mapping settings:

1. In the flow area, select the step for which you want to check the step property values.  
The **Debug** area displays the input properties and output properties of the step you selected.
2. Click the **Service Properties** tab at the bottom of the **Service Builder Debug** window.  
The values of the service properties are displayed.
3. In the **Debug** area and **Service Properties** tab, make sure that the same value appears in the Property Value columns for the step property and the mapped service property.

## Operation results

You can confirm that the property values are inherited as specified in mapping. If a mapped service property is different from the design or if displayed values are not appropriate, you must correct the problem in the **Service Builder Edit** window.

### Important

If you specify `parallel` as the `foreachMode` property of a repeated execution plug-in, values of service properties updated in a repeated execution flow executed in parallel do not appear in the **Service Properties** tab.

## Related topics

- [8.3.9 Operations for interrupting step executions during debugging](#)
- [8.3.16 Displaying step property values and return values during debugging](#)

## 8.3.14 Procedure for changing step property values or return value during debugging

When step execution is interrupted, you can change the values of step properties or return value to any value. If you change an input property value before executing a plug-in, you can see how different property values affect the processing of the plug-in. If you change an output property value or return value after executing or skipping a plug-in, you can see how different property values or return values affect the processing of subsequent steps and the flow transition.

For details about the conditions for changing step property values and return value, see [8.3.10 Step information that can be changed when step execution is interrupted](#).

### To change a step property value or return value during debugging:


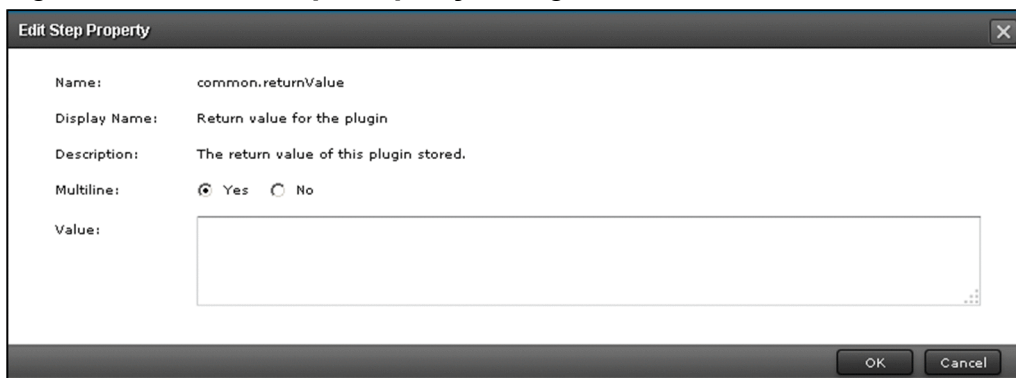
1. In the flow area, select a step that is in Interrupted status.
2. In the **Debug** area, in the **Step Properties** area, click  for the property or return value you want to change.
3. In the **Edit Step Property** dialog box, enter the desired value. When you change the value of a step property, you can select **Yes** for the **Multiline** radio button to set a property value including linefeed.

Figure 8-10: **Edit Step Property** dialog box



4. Click **OK**.

### Operation results

If you perform an operation that executes a step, debug task processing goes on according to the specified step property or return value.



#### Tip

- You can set step property values imported from a property file. You can also export the values specified in the **Service Builder Debug** window to a property file.
- You cannot specify surrogate pair characters or control characters (excluding linefeed and tab characters) for the value of a step property. However, if you have specified a property value containing surrogate pair characters or control characters (excluding linefeed and tab characters) during service template development, that step property value is retained and displayed in the **Debug** area.

### Related topics


- [8.3.15 Importing and exporting step properties in the Service Builder Debug window](#)
- [8.3.9 Operations for interrupting step executions during debugging](#)

- [8.3.17 Effect of changing values of step properties during debugging](#)
- 

## 8.3.15 Importing and exporting step properties in the Service Builder Debug window


You can use the **Service Builder Debug** window to specify the values of input properties of a step in the property file, and output the specified step properties and return values to the property file.

### Importing step properties

In the **Service Builder Debug** window, click **Import** to apply the predefined property file settings to the **Step Properties** area. Property values can be applied to the properties with  displayed in the **Step Properties** area.

When you import properties in the **Service Builder Debug** window, an error occurs if at least one inappropriate value (a value that does not meet the data type or restriction condition) is specified for a property.

If any of the following conditions exists, application of the property value is skipped and then the number of skipped properties is displayed after the import is completed:

- The value of the property cannot be changed (in the **Step Properties** area,  is not displayed for the property).
- The *value* field of the property is not defined in the json-format property file.
- Null is defined in the *value* field of the property in the json-format property file.

If properties that do not exist in a selected step are defined, the number of such properties is displayed after the import is completed.

### Exporting step properties

In the **Service Builder Debug** window, click the **Export** button to output the values of the properties displayed in the **Step Properties** area to the property file.

At this time, the name of the output property file is `step_properties.json`. The return value is output as the `reserved.debugger.exitCode` property key.

Note that when step properties are exported, the *type* field is not output (excluding step properties of service components).

---

### Related topics

- [Overview of property files in the JP1/Automatic Operation Administration Guide](#)
- 

## 8.3.16 Displaying step property values and return values during debugging

The following describes the conditions under which the values of input and output properties of a step appear in the **Step Properties** area during debugging.

Table 8-6: Conditions for displaying step property values

Type	Displayed step	Displayed when:
Input property of a step	Step containing plug-ins that can be interrupted, and branch by property value plug-ins	Plug-in execution begins
Output property of a step	Step containing plug-ins that can be interrupted	Plug-in processing ends
Return value	All steps	Plug-in processing ends

## Related topics

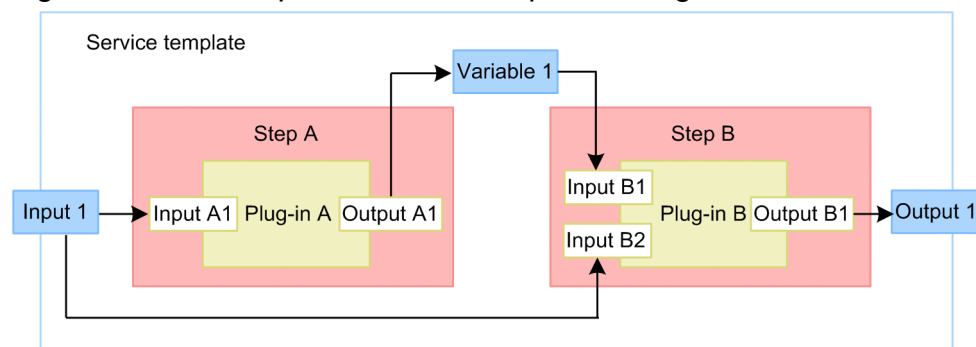
- [8.3.12 Plug-ins that cannot be interrupted or skipped during debugging](#)

## 8.3.17 Effect of changing values of step properties during debugging

During debugging you can change the values of step properties and the return values. If you change the value of a step property, the value of the service property to which the step property is mapped does not change automatically. However, changing an output property of a step might indirectly change the value of a service property (output property or variable) depending on how output property mapping is configured.

The figure below shows an example of the behavior of a service template affected by changing the values of input and output properties of a step while executing a step. This example assumes that the service template is defined as described below.

Figure 8-11: Example of service template configuration



## Property mapping definition

### Step A

- Input property Input 1 of the service is mapped to Input A1 of Step A
- Output property Output A1 of Step A is mapped to Variable 1 of the service.

### Step B

- Variable 1 of the service is mapped to Input B1 of Step B.
- Input 1 of the service is mapped to Input B2 of Step B.
- Output B1 of Step B is mapped to Output 1 of the service.

## Behavior when a property value is changed

### Behavior when changing an input property of a step

If you change the value of Input A1 of Step A before executing Plug-in A, Plug-in A uses the new value when it runs. This does not change the value of the service input property Input 1 to which the step input property Input A1 is mapped. Therefore, the step input property Input B is assigned the original value specified for the input property of the service.


### Behavior when changing an output property of a step

If you change the value of the output property Output A1 of the step after executing Plug-in A, the new value is assigned to Variable 1 to which Output A1 is mapped. The input property Input B1 of the step also takes the new value.

## 8.3.18 Procedure for handling debug tasks that are waiting for a response (response entry)

If a debug task requires a user response during execution, you can enter a response in the **Debug** area.

### To respond to a debug task that is waiting for a user response:

1. Make sure that the debug task is in Waiting for Input status. Then, in the **Debug** area, click the **Enter Response** button (  ).
2. In the **Enter Response** dialog box, confirm the message, and then click the button associated with the action you want to perform.
3. In the **Information** dialog box, click **OK**.

### Operation results

Depending on the response you entered, the plug-in processing of the step resumes or stops.


## 8.3.19 Procedure for debugging a service template again without rebuilding

After executing a debug task, you can debug the same service template again by executing the debug task from the **Service Builder Debug** window displayed. In this case, you do not need to rebuild the service template.

You can rerun debugging if the debug task is in Completed or Failed status. If you want to execute a debug task while another debug task is running, you must forcibly stop the currently running debug task.

When you rerun debugging, JP1/AO deletes the existing debug service and debug task, and then generates a new debug service and debug task.

### To execute a new debug task without rebuilding the service template:

1. In the **Debug** area, from the **Resubmit** pull-down menu (  ), select **Retry the Task**.
2. In the **Perform Debugging** dialog box, specify the definition information for the debug service and debug task, task log output level, and debug service property information.

3. Click **OK**.

## Operation results

The debug task is executed.

### Important

- The **Perform Debugging** dialog box displays the service template as it was configured when last built by the user who performs debugging. If another user edits and builds the same service template in the meantime, the changes do not apply to the contents of the service template displayed in this dialog box.
- You can debug a service template again without rebuilding it only while the **Service Builder Debug** window is displayed. If you close the **Service Builder Debug** window by clicking the **Back to Editor** button or logging out of JP1/AO, you need to rebuild the service template before debugging it.

---

## Related topics

- [8.3.2 General procedure for debugging service templates](#)
  - [8.3.6 Settings used when starting debugging](#)
  - [8.4.5 Procedure for forcibly stopping debug tasks](#)
- 


## 8.3.20 Procedure for retrying a task from a failed step during debugging

If a debug task fails partway through, you can retry the task from the failed step.

By retrying 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 has been resolved. For example, a step that fails due to a temporary problem with the network can be retried when the network connection is available again.

For details about retrying tasks, such as exceptions when property values are inherited and the possibility of retry depending on the status of a debug task, see *Retrying tasks* in the *JP1/Automatic Operation Administration Guide*.

### To retry a task from a failed step during debugging:

1. Make sure that the debug task is in Failed status. Then, in the **Debug** area, from the **Resubmit** pull-down menu (  ), select **Retry From Failed Step**.

A dialog box appears in which you can confirm that you want to retry the task from the failed step.

2. Click **OK**.

## Operation results

The task is re-executed from the failed step.

---

## Related topics

- [8.3.21 Procedure for retrying a task from the step after the failed step during debugging](#)
  - [Retrying tasks in the JP1/Automatic Operation Administration Guide](#)
-


### 8.3.21 Procedure for retrying a task from the step after the failed step during debugging

When a debug task fails partway through, you can retry the task 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 appropriate in situations where there is no need to execute the failed step. When you retry a task from the step after the failed step, processing of the task continues as if the failed step had ended normally. You can use this approach when you find a problem in a step, but want to continue executing the debug task and deal with the problem later.

For details about retrying tasks, such as exceptions when property values are inherited and the possibility of retry depending on the status of a debug task, see *Retrying tasks* in the *JP1/Automatic Operation Administration Guide*.

#### To retry a task from the step after a failed step during debugging:

1. Make sure that the debug task is in Failed status. Then, in the **Debug** area, from the **Resubmit** pull-down menu (  ), select **Retry From Next To Failed Step**.

A dialog box appears in which you can confirm that you want to retry the task from the step after the failed step.

2. Click **OK**.

#### Operation results

The task is re-executed from the failed step.

---



#### Related topics

- [8.3.20 Procedure for retrying a task from a failed step during debugging](#)
  - [Retrying tasks in the JP1/Automatic Operation Administration Guide](#)
- 

### 8.3.22 Displaying debug task flow

You can display a flow that represents the debug task you are executing.

In the flow area in the **Service Builder Debug** window, the steps in the debug task appear in the order in which they are executed.

Step icons include the icon that indicates the status of a step and the breakpoint icon (  ). If the step contains a flow plug-in, repeated execution plug-in, or a service component, an arrow icon (  ) appears at the lower right of the step icon. Clicking the arrow icon displays the subordinate flow.

When you rest your mouse pointer on the step icon, the step name and the status of the step are displayed. For executed steps, the start time, end time, and return value of the step are also displayed.

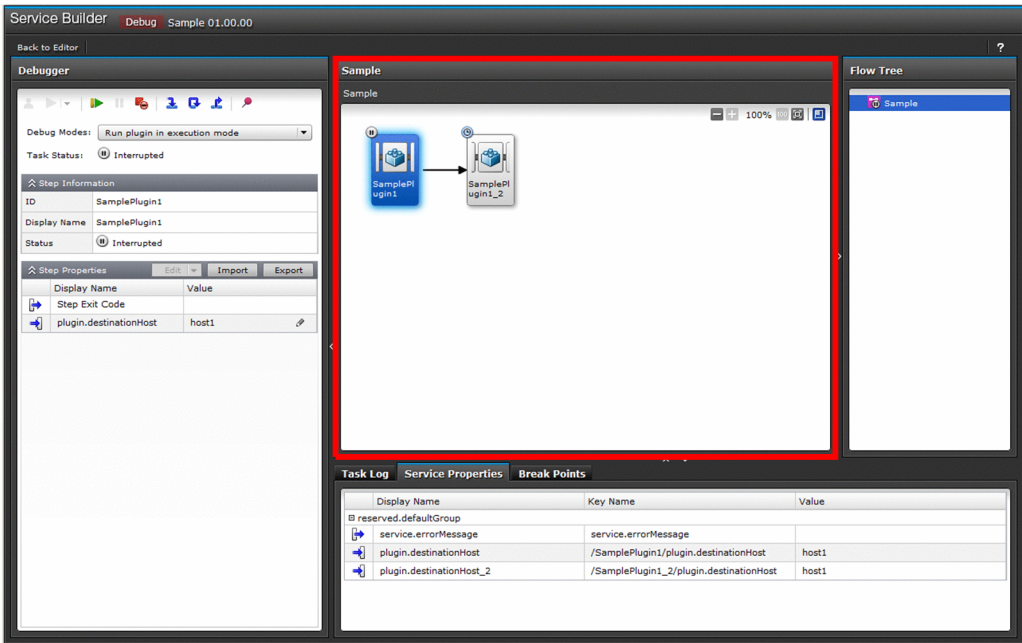


#### Tip

For details about step statuses, see *Step statuses* in the *JP1/Automatic Operation Administration Guide*.



Figure 8-12: Displaying the flow of a debug task



In the flow area, the border around the icon of an interrupted step is highlighted as follows.

Figure 8-13: Icon of interrupted step (example)



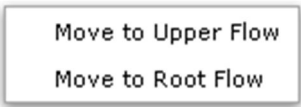
In the flow area, steps with plug-ins that cannot be interrupted have an icon with a darker background as shown below.

Figure 8-14: Example icon of steps with plug-ins that cannot be interrupted and steps in subordinate flows of Repeated Execution plug-ins (before execution)



If the service template has a hierarchical structure, right-clicking the darker background of the image showing the steps in a subordinate flow displays the following menu, which allows you to move to the upper flow or root flow.

Figure 8-15: Menu displayed by right-clicking the darker background of the image showing steps in a subordinate flow



Right-clicking a step icon displays the following menu, which allows you to perform the step execution action, display the task log, and select **View Details** to open the **View Step Attributes** dialog box. For a flow plug-in, Repeated Execution Plug-in, or service component, selecting **Open** displays the subordinate flow.

Figure 8-16: Menu displayed by right-clicking a step icon

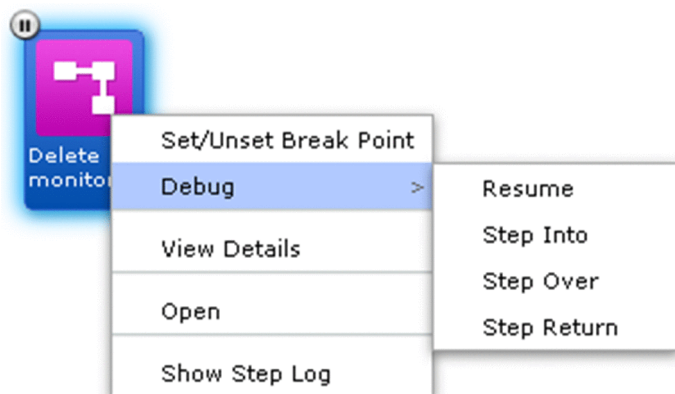
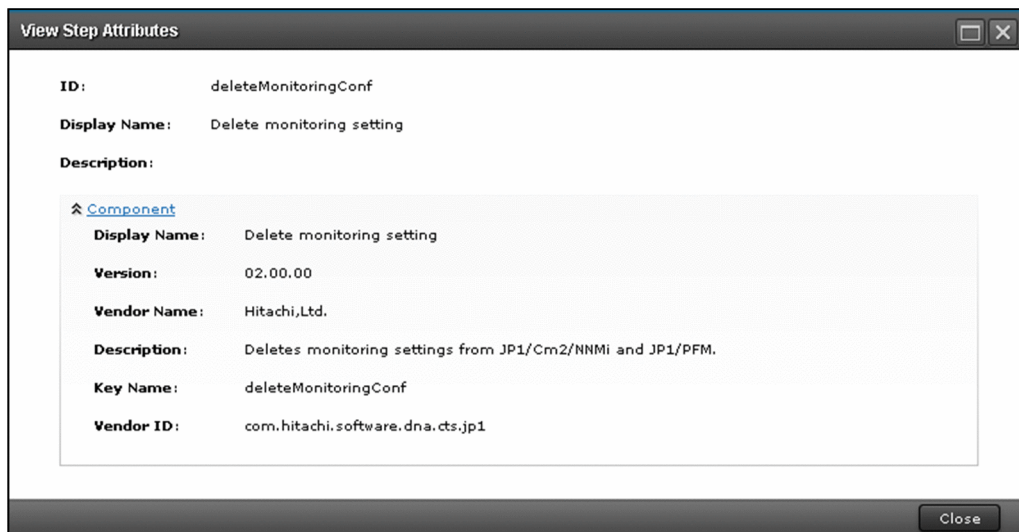


Figure 8-17: View Step Attributes dialog box



## Related topics

- [8.3.24 Displaying a repeated execution flow during debugging](#)
- [8.3.25 Information displayed for repeated execution plug-ins and repeated execution flows during debugging](#)

## 8.3.23 Displaying the flow tree of a debug task

The name of the service template appears at the top level of the flow tree. Lower levels are represented by the name of the step that executes the flow plug-in, Repeated Execution Plug-in, or service component.

During debugging, an interruption icon is displayed for hierarchical levels whose steps are interrupted (in a status requiring user intervention).

Figure 8-18: Example of interruption icon (when a step in Flow plug-in 2 is interrupted)



To find interrupted steps during debugging, look for hierarchical levels with interruption icons in the flow tree. By selecting a level with the interruption icon, you can identify the interrupted step from the flow that appears.

Interruption icons are not displayed for levels that are above the level containing the interrupted step in the hierarchy. Note that the status of the step that represents the flow plug-in or Repeated Execution Plug-in itself and the status of the repeated execution flow are not displayed in the flow tree.

---

## Related topics

- [8.3.24 Displaying a repeated execution flow during debugging](#)
  - [8.3.25 Information displayed for repeated execution plug-ins and repeated execution flows during debugging](#)
- 

### 8.3.24 Displaying a repeated execution flow during debugging

For steps that execute a Repeated Execution Plug-in during debugging, a flow for each iteration is displayed in the level below the Repeated Execution Plug-in. This is called a repeated execution flow. Repeated execution flows appear in the flow area and the **Flow Tree** area as soon as execution of a Repeated Execution Plug-in starts (and remain displayed during and after execution of the plug-in).

The name of a repeated execution flow is displayed in Step *[iteration-number]:input-value-(reserved.loop.input)* format. Displayed repeated execution flow names that are 65 characters or longer are truncated after the 64th character. If the Input Properties value contains control characters, the name is truncated after the 64th character after removing the control characters. Note that the *iteration-number* is a two-digit number.

The following shows an example of a Repeated Execution Plug-in executed when *hostA, hostB, hostC* is specified for the input property of the Repeated Execution Plug-in (inputProperties). In this example, the displayed flow names are Step[01]:hostA, Step[02]:hostB, and Step[03]:hostC.

Figure 8-19: Flow display for repeated execution flows

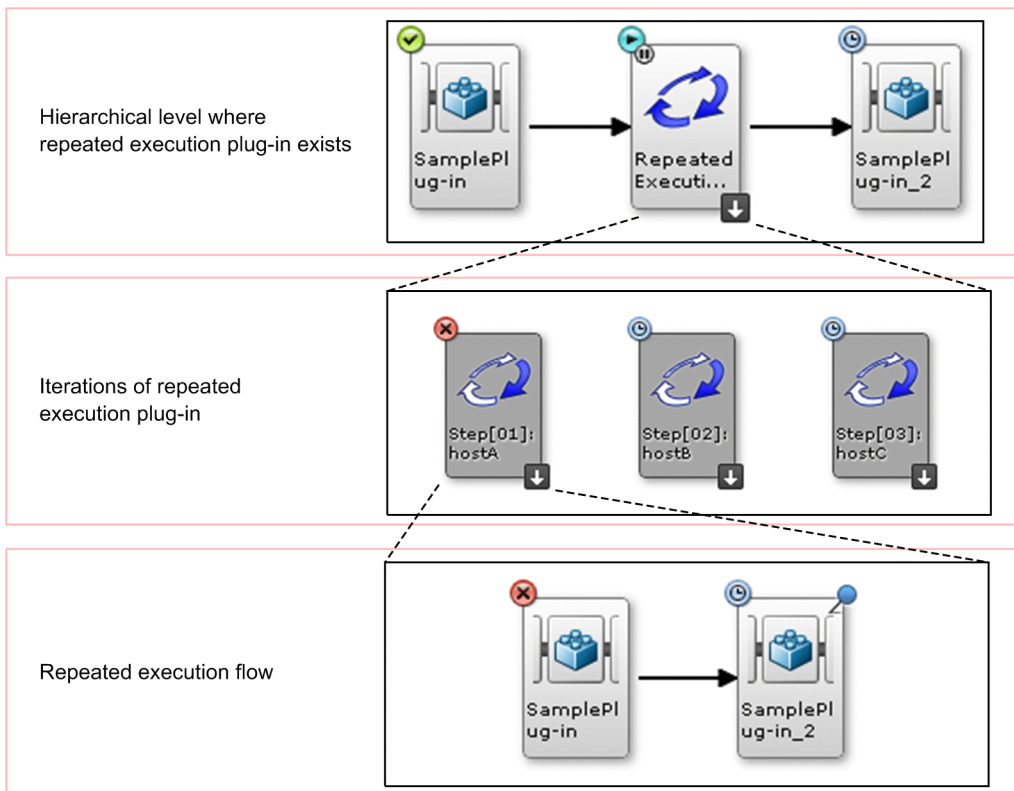
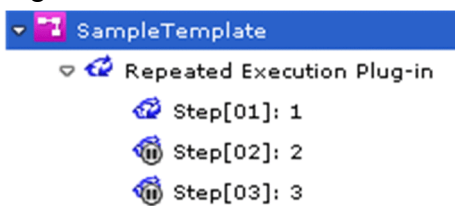


Figure 8-20: Flow tree view of repeated execution flows



Note that repeated execution flows for all iterations are displayed regardless of whether the repeated execution method is *parallel* or *serial*.

However, if you log out during debugging, information about iterations might no longer be displayed for repeated execution flows that are in Waiting for Loop Execution status.

If you set breakpoints for a step in the subordinate flow before executing the Repeated Execution Plug-in, the breakpoint settings are applied to the same step in the repeated execution flow displayed after the processing is performed. If you set or cancel breakpoints for a step in a repeated execution flow after repeated execution flows are displayed, the breakpoint settings are applied to the same step in other repeated execution flows.

## Related topics

- [8.3.24 Displaying a repeated execution flow during debugging](#)
- [8.3.23 Displaying the flow tree of a debug task](#)
- [8.3.25 Information displayed for repeated execution plug-ins and repeated execution flows during debugging](#)

## 8.3.25 Information displayed for repeated execution plug-ins and repeated execution flows during debugging

The table below describes the information displayed for repeated execution plug-ins and repeated execution flows when a step that includes a repeated execution plug-in is executed during debugging. This information does not appear until the step has started executing.

Table 8-7: Information displayed for repeated execution flows

Item	Description
Repeated execution method (parallel or serial)	The value of the <code>foreachMode</code> property of the step that contains the Repeated Execution Plug-in is displayed.
Iteration number ( <code>reserved.loop.index</code> , <code>reserved.loop.indexN</code> )	<p>This information is combined with the applicable Input Properties value to give the name of the repeated execution flow.</p> <p>For the comma-separated values specified in the <code>inputProperties</code> property for the step that contains the Repeated Execution Plug-in, this information indicates the position (number) of the parameter that corresponds to the current execution. For details about the reserved properties <code>reserved.loop.index</code> and <code>reserved.loop.indexN</code>, see the topic <a href="#">3.6.8 List of reserved properties</a>.</p>
Input Properties ( <code>reserved.loop.input</code> , <code>reserved.loop.inputN</code> )	<p>This information is combined with the iteration number to give the name of the repeated execution flow.<sup>#</sup></p> <p>For the comma-separated values specified in the <code>inputProperties</code> property for the step that contains the Repeated Execution Plug-in, this information indicates the value that corresponds to the current iteration of the flow. For details about the reserved properties <code>reserved.loop.input</code> and <code>reserved.loop.inputN</code>, see the topic <a href="#">3.6.8 List of reserved properties</a>.</p>
Repeated execution result (true or false)	The value of the <code>outputResult</code> property for the step that contains the Repeated Execution Plug-in is displayed.
Output value ( <code>reserved.loop.output</code> )	<p>The value of the <code>outputProperties</code> output property for the step that contains the Repeated Execution Plug-in is displayed.</p> <p>This value has been specified for <code>reserved.loop.output</code> by output property mapping in the repeated execution flow.</p>

#

Repeated execution flow names that are 65 characters or longer are truncated after the 64th character. If the Input Properties value contains control characters, names are truncated after the 64th character after removing the control characters.

## 8.4 Managing debug tasks

---

You can perform the following operations for debug tasks in addition to execution:

- Check the progress
- Check detailed information
- Check and download the task log
- Stop execution
- Forced stop
- Deletion

### 8.4.1 Procedure for checking progress of debug tasks from the Tasks window

From the **Tasks** window, you can view the progress of debug tasks as a flow. For example, you use the **Tasks** window to check the status of steps in a debug task after closing the **Service Builder Debug** window, or check the status of a step in a debug task executed by another user.

#### To view the progress of a debug task from the Tasks window:

1. In the **Tasks** window, click the **Debug** tab.

For details about how to check the progress of debug tasks from the **Tasks** window, see *Checking task statuses from the **Tasks** window* in the *JP1/Automatic Operation Administration Guide*.

#### Important

JP1/AO automatically deletes debug tasks (and archives tasks) whose retention period has expired and tasks exceeding the total number of tasks (including normal tasks) that can be retained. This takes place once a day at the same time as the automatic archiving of normal tasks. Tasks are deleted or archived by date from the task with the oldest end date. You can change the retention period for debug tasks, the total number of tasks that can be retained, and the timing of automatic deletion in the user-specified properties file (config\_user.properties). For details about the automatic archiving of tasks, see *Automatically archiving tasks and deleting task histories* in the *JP1/Automatic Operation Administration Guide*.

### Operation results

You can check the progress of debug tasks.

---

#### Related topics

- [8.3.23 Displaying the flow tree of a debug task](#)
  - [8.3.24 Displaying a repeated execution flow during debugging](#)
  - User-specified properties file (config\_user.properties) in the JP1/Automatic Operation Configuration Guide
  - Step statuses in the JP1/Automatic Operation Administration Guide
-

## 8.4.2 Procedure for checking details about debug tasks from the Task Details window

In the **Task Details** window, you can check information such as the property values specified for the service during debugging and the task log. For example, you can use the **Task Details** window to check the information after closing the **Service Builder Debug** window, or check details about a debug task executed by another user.

### To check details about debug tasks from the Task Details window:

1. In the **Tasks** window, click the **Debug** tab.

For information on how to check details about debug tasks from the **Tasks** window, see *Viewing detailed task information* in the *JPI/Automatic Operation Administration Guide*.

## Operation results

You can check details about the debug tasks.

## 8.4.3 Procedure for checking task log entries for debug tasks

The following describes how to view the information output to the task log by a debug task. You cannot view the task log for a debug task that is in Waiting status. You can also download the task log to a folder of your choice, under any file name.

You can view the task log for a running debug task from the **Service Builder Debug** window.

To view the task log for a finished debug task or a debug task that was executed by another user, use the **Tasks** window.

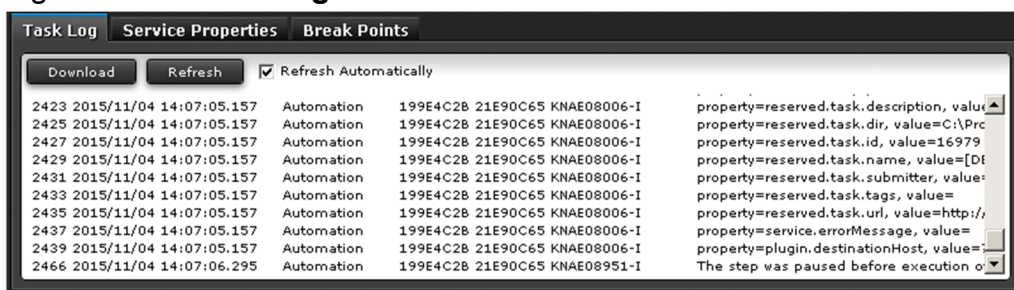
For details about the information output to the task log, see *Task log details* in the *JPI/Automatic Operation Administration Guide*.

### To view the task log for a debug task from the Service Builder Debug window:

1. In the **Service Builder Debug** window, click the **Task Log** tab.

The contents of the task log of the debug task you are debugging appear on the **Task Log** tab.

Figure 8-21: Task Log tab



You can use the **Task Log** tab to perform actions such as setting the timing with which the contents of the task log are refreshed, and downloading the task log information to a file. The following describes how to perform these actions.

- To download the task log:  
Click **Download**.
- To manually update the task log:



Click **Update**.

- To automatically update the task log at regular intervals:

Select the **Refresh Automatically** check box.

When you select this check box, the task log is automatically updated each time the status of a step or task changes while the debug task is running.

### Tip

- If you right-click a step icon in the flow area and then select **Show Step Log**, the task log automatically scrolls up to the location where the result of the step is displayed.
- When you retry a debug task or debug a debug task that has already been debugged, the **Refresh Automatically** check box is selected automatically and the task log is updated.
- If the **Refresh Automatically** check box is selected, regardless of the task status, the task log is refreshed and the scroll box on the **Task Log** tab scrolls to the bottom of the table.

## To view the task log for a debug task from the Tasks window:

1. In the **Tasks** window, click the **Debug** tab.

For details about how to check the task log from the **Tasks** window, see *Viewing detailed task information* in the *JP1/Automatic Operation Administration Guide*.

### Important

- The task log size specified (in KB) in the property file (config\_user.properties) determines the amount of information displayed in the **Task Log** tab in the **Task Details** window and **Service Builder Debug** window. This amount (in KB) of information is taken from the end of the task log data. The dialog boxes do not display the entire contents of the task log.
- You can set the maximum log file size for debug tasks (in KB) in the user-specified properties file (config\_user.properties). If the maximum file size is exceeded, JP1/AO begins overwriting the oldest information in the task log.

---

## Related topics

- User-specified properties file (config\_user.properties) in the JP1/Automatic Operation Configuration Guide
- 

## 8.4.4 Procedure for stopping debug tasks

The following describes how to stop a debug task that is still in progress. You can stop debug tasks that are in the status of In Progress, Waiting for Input, or In Progress (with Error). Stopped tasks enter Failed status. However, if the user stops a debug task while the final step is in progress, the debug task enters Completed status if the final step ends normally.

If the debug task contains an interrupted step, the step is automatically resumed and then the debug task stops when the step finishes executing.





### Tip

For details about the difference between stopping and forcibly stopping tasks, see *Managing Tasks* in the *JPI/Automatic Operation Administration Guide*.

## To stop debug tasks:

1. In the **Tasks** window, click the **Debug** tab.
2. In the list of tasks, select the debug tasks that you want to stop.



### Tip

You can select multiple debug tasks by selecting check boxes beside the task names.

3. In the **More Actions** pull-down menu, click **Stop Task**.
4. In the **Stop Task** dialog box, confirm the debug tasks to be stopped, and then click **OK**.
5. In the **Information** dialog box, click **OK**.

## Operation results

The selected debug tasks enter Failed status.

## 8.4.5 Procedure for forcibly stopping debug tasks

The following describes how to forcibly stop a debug task that is still in progress. You can forcibly stop debug tasks that are in the status of In Progress, Waiting for Input, In Progress (with Error), or In Progress (Terminating).

If you want to stop a running debug task and perform the debug process again, you can forcibly stop the debug task from the **Service Builder Debug** window.

If you inadvertently close the Web browser window from which you are using JPI/AO while a debug task is in progress, use the **Tasks** window to forcibly stop the debug task. You can also use this window to forcibly stop a debug task executed by another user.

Debug tasks that are forcibly stopped enter Failed status.


If the debug task contains an interrupted step, the debug task stops after the step is automatically resumed, without executing the plug-in processing.



### Tip

For details about the difference between stopping and forcibly stopping tasks, see *Managing Tasks* in the *JPI/Automatic Operation Administration Guide*.

## To forcibly stop a debug task from the Service Builder Debug window:

1. In the **Debug** area of the **Service Builder Debug** window, click the **Forcibly Stop** button (  ).
2. In the **Information** dialog box, click **OK**.

## Operation results

The debug task enters Failed status.

### To forcibly stop debug tasks from the Tasks window:

1. In the **Tasks** window, click the **Debug** tab.
2. In the list of tasks, select the debug tasks that you want to forcibly stop.



#### Tip

You can select multiple debug tasks by selecting check boxes beside the task names.

3. In the **More Actions** pull-down menu, click **Forcibly Stop**.
4. In the **Forcibly Stop** dialog box, confirm the debug tasks to be stopped forcibly, and then click **OK**.
5. In the **Information** dialog box, click **OK**.

## Operation results

The selected debug tasks enter Failed status.

## 8.4.6 Procedure for deleting debug tasks

The following describes how to manually delete debug tasks that are no longer required. You can delete debug tasks that are in Completed or Failed status.

If you want to delete debug tasks automatically, set a retention period for executed debug tasks in the user-specified properties file (config\_user.properties).

### To manually delete debug tasks:

1. In the **Tasks** window, click the **Debug** tab.
2. In the list of tasks, select the debug tasks that you want to delete.



#### Tip

You can select multiple debug tasks by selecting check boxes beside the task names.

3. In the **More Actions** pull-down menu, click **Delete Tasks**.
4. In the **Delete Tasks** dialog box, confirm the debug tasks to be deleted, and then click **OK**.

## Operation results

The debug tasks are deleted from the **Debug** tab.

---

### Related topics

- User-specified properties file (config\_user.properties) in the JP1/Automatic Operation Configuration Guide
-

## 8.5 Testing the operation of service templates

---

When you have finished debugging a service template, you can perform an operation test before releasing the service template.

### 8.5.1 Procedure for testing the operation of service templates

When you have finished debugging the service template and make sure that no problems remain, you can perform an operation test in the development environment.

#### To test the operation of a service template:

1. After building the service template, create a service in the **Services** window.



#### Tip

In the **Select Service Template** dialog box, click the **Show All Versions** button to display the development service templates that have been built. Click the **Show Latest Version** button to display only the release service template.

2. Execute the service.
3. Check the execution results in the **Tasks** window.
4. If the results of the operation test reveal a problem with the service template, edit the affected service template or plug-in.

---

#### Related topics

- [Creating services in the JP1/Automatic Operation Administration Guide](#)
  - [Executing Services in the JP1/Automatic Operation Administration Guide](#)
  - [Managing Tasks in the JP1/Automatic Operation Administration Guide](#)
  - [8.1.1 General procedure for validating service templates](#)
  - [8.1.2 Overview of building](#)
  - [8.1.4 Overview of operation tests](#)
  - [8.2.1 Procedure for building service templates](#)
-

# Appendix

## A. Reference Information

---

This appendix provides reference information for users of JP1/AO.

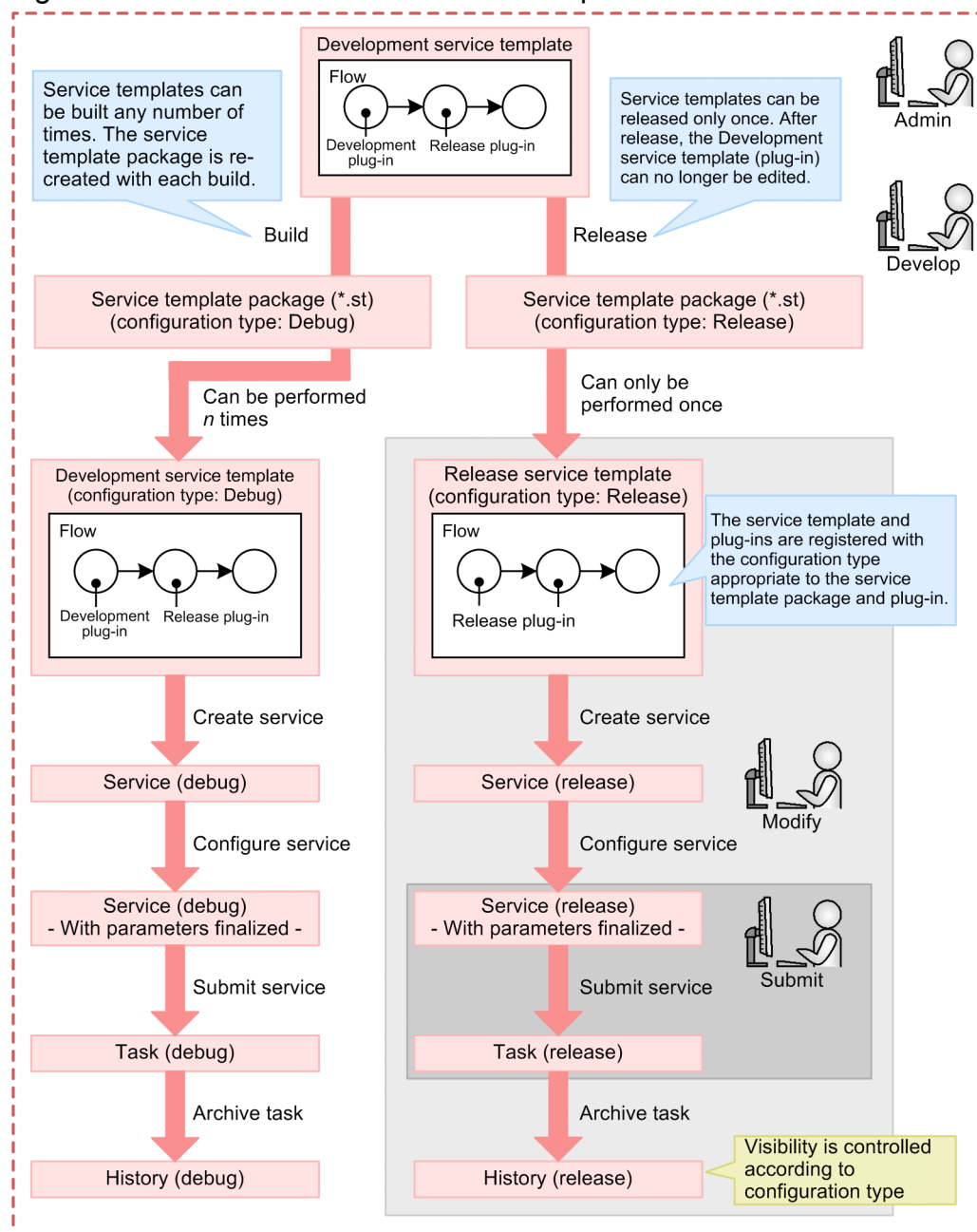
### A.1 Reference information for build and release operations

#### (1) Structure of build and release processes and how they differ

The build and release processes create a package of a service template. Perform a build operation when you want to validate a service template, and a release operation when you want to make the service template available in the active environment.

The following figure shows the structure of the build and release processes.

Figure A-1: Structure of build and release processes



The following table lists the differences between build and release processes.

Table A-1: Differences between build and release

Operation	Objective	Number of executions	Configuration type assigned to service template package	File name of service template package	Can development service template be edited after operation?
Build	Validate a service template	Any number of times	Debug configuration	<i>vendor-ID_name_version_d.st</i>	Yes
Release	Make a service template available in the active environment	Once	Release configuration	<i>vendor-ID_name_version.st</i>	No

Legend:

Yes: Can be edited. No: Cannot be edited.

---

## Related topics

- (2) Configuration types assigned to service templates and plug-ins by build and release operations
  - (4) Ability to display window items by service template configuration type and the user role
  - (5) Deletion and archiving of service templates, services, and tasks during build and release operations
- 

## (2) Configuration types assigned to service templates and plug-ins by build and release operations

JP1/AO assigns a configuration type to the service templates and plug-ins in a service template package (\*.st) according to its stage in the development process (build or release). The following table lists the configuration types assigned to service templates and plug-ins.

Table A-2: Configuration types assigned by build and release operations

Service template or plug-in		Assigned configuration type	
		When built	When released
Development service template		Debug configuration	Release configuration
Plug-in <sup>#</sup>	Basic plug-in or release plug-in (release configuration)	Release configuration	Release configuration
	Development plug-in (debug configuration)	Debug configuration	Release configuration

#

You cannot build or release individual plug-ins. A development plug-in becomes a release plug-in when you release a service template that contains that plug-in. A development plug-in remains as such when you build a service template that contains the plug-in.

---

## Related topics

- (4) Ability to display window items by service template configuration type and the user role
- 

## (3) Build or release operations performed on multiple instances of the same service template or plug-in

If you repeatedly build a service template during its development, or you build then later release a service template, the same service template or plug-ins might have already been imported. The behavior of the build or release operation in this scenario depends on the configuration type of the service template or plug-in.

Note that the same service template (or same plug-in) means a service template or plug-in with the same vendor ID, service template ID (plug-in ID), and service template version (plug-in version). The plug-ins referred to below are those used by the service template being built or released.

### When building a service template

- The service template is overwritten.
- Development plug-ins in the service template you are building are overwritten.

- Release plug-ins in the service template you are building are not overwritten.

## When releasing a service template

- The service template is overwritten and becomes a release service template.
- Development plug-ins in the released service template become release plug-ins.
- Release plug-ins in the released service template are not overwritten.

## (4) Ability to display window items by service template configuration type and the user role

The configuration type of the service template and the role of the user determine whether service templates and elements (services, tasks, and histories) created from service templates appear in the **Services** window and the **Tasks** window. The following table describes whether these items appear in the user interface for each combination of configuration type and user role.

Table A-3: Ability to display window items by service template configuration type and user role

Configuration type of the service template	Managed element	Displayed in the Services window and Tasks window		
		Admin Develop	Modify	Submit
Debug configuration	Service template	Y	N	N
	Service	Y	N	N
	Task	Y	N	N
	History	Y	N	N
Release configuration	Service template	Y	Y	N
	Service	Y	Y	Y
	Task	Y	Y	Y
	History	Y	Y	VO

Legend:

Y: Can be viewed and manipulated. VO: Can only be viewed. N: Cannot be viewed or manipulated.

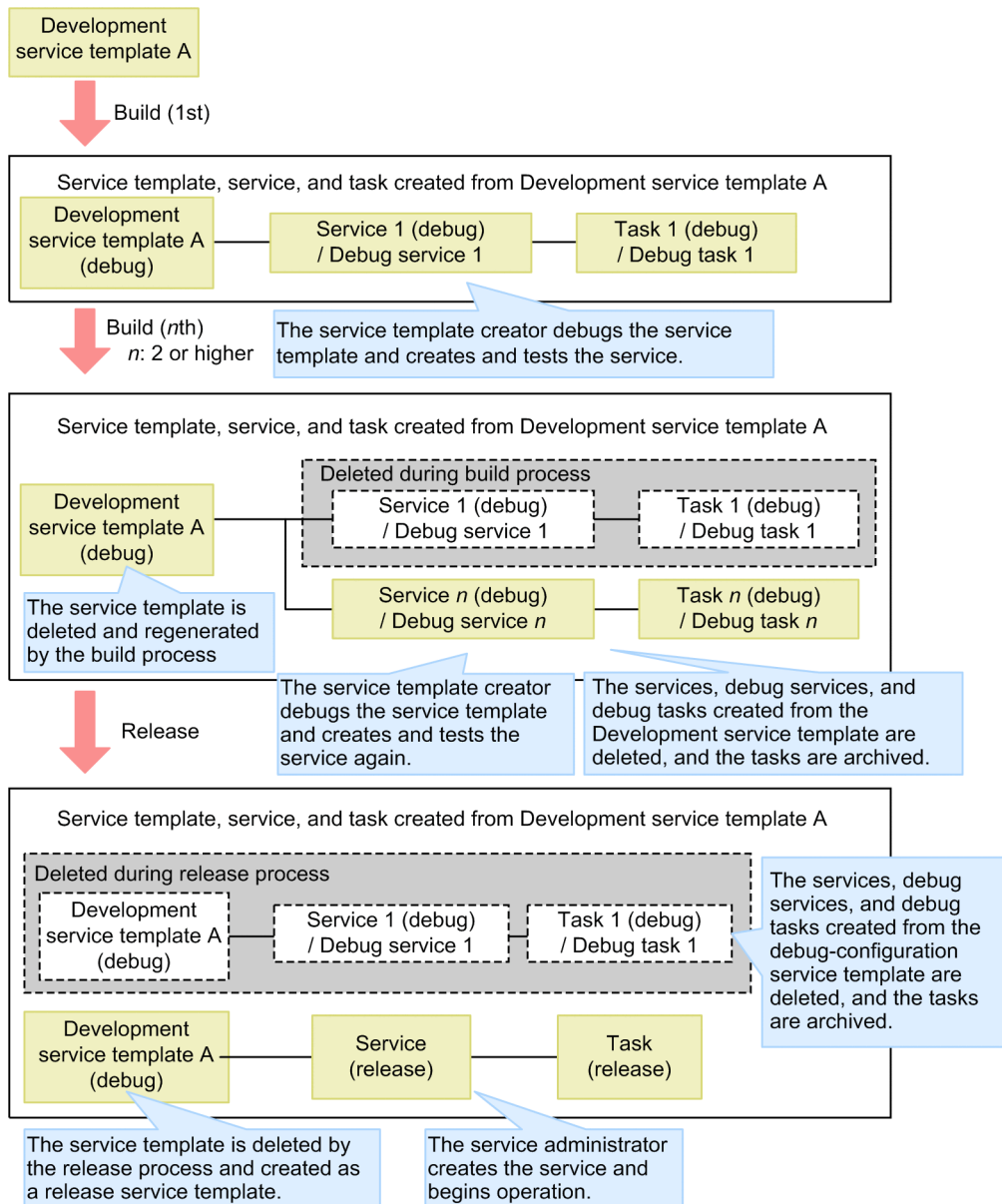
## (5) Deletion and archiving of service templates, services, and tasks during build and release operations

When you build or release a development service template, JP1/AO automatically deletes service templates and services and archives tasks. Debug services and debug tasks are automatically deleted.

Note that service templates and services are deleted and tasks archived even if the build or release process fails. Debug services and debug tasks are also deleted.



Figure A-2: Deletion and archiving of elements during build and release operations



## A.2 Compatibility for service templates

This section describes the compatibility with earlier or later versions of JP1/AO.

### (1) Compatibility for service templates and plug-ins created in the Service Builder window

The following describes the compatibility for service templates and plug-ins created in the **Service Builder**.

#### Compatibility with later versions

Service templates are guaranteed to remain compatible with later versions. This means that you can work with service templates that meet the conditions below in the **Service Builder** window. However, some displayed items and operations might differ between versions.

- Service template packages (\*.st) created by users in earlier versions

- Service templates and plug-ins imported to the JP1/AO server from an earlier version of JP1/AO

#### Compatibility with earlier versions

Service templates are not guaranteed to be backwards compatible. If you attempt to apply a service template created in the **Service Builder** window to an earlier version of JP1/AO, an error might occur during the import process.

#### Compatibility with JP1/AO servers with different operating systems

User-created service templates can be imported to JP1/AO servers with different operating systems. However, the following restrictions apply:

- Only ASCII characters can be used for default values of input properties of plug-ins, and the values of input properties for plug-ins (or steps) in a flow.
- Only service templates created in JP1/AO 10-10 or later can be imported to a JP1/AO server that is running Linux.

## (2) Compatibility with steps in service templates created in earlier versions of JP1/AO

Some steps in service templates created in versions of JP1/AO earlier than 10-10 are not compatible with JP1/AO 10-10. This might prevent you from changing the processing in the step. The icon for steps that are incompatible with JP1/AO 10-10 is displayed in gray scale as shown below.

Figure A-3: Icon for incompatible steps



- You cannot change the processing defined in a step. You can only change the step ID, step name, and description.
- The default step ID is compatible.step.

If there are multiple incompatible steps at the same hierarchical level, *\_n* is appended to the end of the step ID (where *\_n* is a unique integer greater than or equal to 2) to give a step ID in *step-ID\_n* format. Step IDs are automatically assigned by the system. The value of *\_n* has no relationship with the position of the step in the flow.

- The description of a step is initially a null character.

## A.3 Version changes

### (1) Changes in version 11-51

In the **Specify Component Input Property Mapping Parameters** dialog box or the **Specify Component Output Property Mapping Parameters** dialog box, by selecting the **View Property** radio button, you can now display only the properties that can be mapped. As such, the relevant descriptions were changed.

### (2) Changes in version 11-50

- Descriptions of screen displays and procedures related to steps were changed.
- The descriptions of the following shared built-in service properties were deleted:
  - com.hitachi.software.dna.sys.jp1.username

- com.hitachi.software.dna.sys.jp1.password
- The supplementary explanation for the procedure for importing service templates was deleted.
- Notes on command execution were added.

### (3) Changes in version 11-10

- The **Service Builder** window was redesigned.
- Available Actions was added to the items that can be specified in the **Edit Service Template Attributes** dialog box.
- A description about the **Next Steps** tab that is displayed in the **Step Properties** area was added.
- The description about repeated steps was changed. A nested structure, where a maximum of three hierarchical levels of repeated steps can be used, is now available.
- The description was changed because the labels for the conditions displayed in the **Determine the return value based on the threshold > Condition** pull-down menu in the **Create Step** window and the **Edit Step** window were changed.
- The next step conditions can now be specified for flow plug-in, and content indicating otherwise was deleted.
- A description about conditional expressions that use arrows to indicate connections with subsequent steps was added.
- Auto-completion of property values is now available for the flow plug-in, interval plug-in, branch by property value plug-in, and abnormal-end plug-in, and content indicating otherwise was deleted.
- A description about the **RESERVED PROPERTY** dialog box, which displays the reserved properties related to repeated execution plug-ins, was added.
- The reserved properties reserved.loop.indexN and reserved.loop.inputN were added.
- A description about the **Select Reference Property** window was added for cases where repeated execution plug-ins are defined in a nested structure.
- Because JP1/AJS3 is no longer bundled with JP1/AO, descriptions related to return values were deleted.
- Regarding the interval plug-in, branch by returncode plug-in, abnormal-end plug-in, and branch by property value plug-in, the specifications concerning the availability of interruption and skipping during debugging were changed. The manual was revised accordingly.
- The reserved properties reserved.loop.indexN and reserved.loop.inputN were added to the information displayed for repeated execution flows.

### (4) Changes in version 11-01

- A note about the file sizes of imported service templates was added.
- An explanation was added noting that if you enable the **Run as system account(Windows)** option when creating or editing a plug-in, commands and scripts will be executed on the connection destination host with the permissions of the System account.
- The description of the information displayed in a flow was changed.
- A cautionary note regarding the procedure for starting debugging was changed.
- Changes were made to the step information that can be changed when step execution is interrupted.

### (5) Changes in version 11-00

#### (a) Changes from the manual (3021-3-087-40)

- The following operating systems are now supported:

- Linux 7
- Oracle Linux 6 (x64)
- Oracle Linux 7
- CentOS 6 (x64)
- CentOS 7
- SUSE Linux 12
- The following operating systems are no longer supported:
  - Linux 5 (AMD/Intel 64)
  - Linux 5 Advanced Platform (AMD/Intel 64)
- The product was migrated from 32-bit Windows to 64-bit Windows.
- The installation folder was changed for the Windows version of JP1/AO and the Common Component.
- A description of using JP1/AO in English and Chinese-language environments was added.
- The structure and contents of the manual were changed to reflect the redesign of the JP1/AO interface.
- A function was added that exports service templates.
- A function was added to place a release service template as a service component in a flow of a service template. Accordingly, the existing plug-ins and service components are now collectively called components.
- Descriptions about visibility and display settings for service properties were added.
- Functions were added to specify visibility and display settings for service properties and the property group to which the service properties belong.
- A function was added to map step properties.
- Auto-completion of property values was added.
- A function was added to specify an image file for service overview as a custom file.
- A function was added that sets the following items in the service resource file:
  - Service property name
  - Service property description
  - Step property name
  - Step property description
- A description of property mapping was added.
- Descriptions about whether properties can be mapped depending on the visibility or data type were added.
- A function was added that elevates step properties to service properties.
- *Tag management* was added as a way to classify service templates and services. Accordingly, category management was removed as a classification method.
- Service groups were added as a way to manage resources. Accordingly, resource groups were removed.
- A function was added that imports and exports service properties.
- Step-into, step-over, step-return, and breakpoint setting were added as the functions for executing a debug task during debugging.
- The following were added as the causes in the description of return value 71 of a content plug-in:
  - An attempt to move to the execution directory failed.
  - An attempt to set an environment variable failed.

- *The total data output to the standard output and standard error output exceeds 100KB* was added to the description of return value 72 of a content plug-in.
- The name of a basic plug-in was changed from File-Forwarding Plug-in to File-Transfer Plug-in.
- The name of a basic plug-in was changed from Judge ReturnCode Plug-in to Branch by ReturnCode Plug-in.
- The name of a basic plug-in was changed from Judge Value Plug-in to Branch by Property Value Plug-in.

## **(b) Changes from the manual (3021-3-363-10(E))**

- Linux was added as a supported operating system.
- The installation folder was changed for the Windows version of JP1/AO and the Common Component.
- The product was migrated from 32-bit Windows to 64-bit Windows.
- The structure and contents of the manual were changed to reflect the redesign of the JP1/AO interface.
- A procedure for changing a plug-in version was added as part of the addition of the plug-in version management function.
- A description of the local execution function was added. This function allows users to start processes directly on local hosts and perform tasks such as executing commands and copying files.
- Keyboard interactive authentication was added as an authentication method used for SSH connections with operation target devices.
- Service groups were added as a way to manage resources. Accordingly, resource groups were removed.
- A function was added that exports service templates.
- A function was added to place a release service template as a service component in a flow of a service template. Accordingly, the existing plug-ins and service components are now collectively called components.
- Descriptions about visibility and display settings for service properties were added.
- Functions were added to specify visibility and display settings for service properties and the property group to which the service properties belong.
- A function was added to map step properties.
- Auto-completion of property values was added.
- A function was added to specify an image file for service overview as a custom file.
- A function was added that sets the following items in the service resource file:
  - Service property name
  - Service property description
  - Step property name
  - Step property description
- A description of property mapping was added.
- Descriptions about whether properties can be mapped depending on the visibility or data type were added.
- A function was added that elevates step properties to service properties.
- *Tag management* was added as a way to classify service templates and services. Accordingly, category management was removed as a classification method.
- A function was added that imports and exports service properties.
- Step-into, step-over, step-return, and breakpoint setting were added as the functions for executing a debug task during debugging.

- The following were added as the causes in the description of return value 71 of a content plug-in:
  - An attempt to move to the execution directory failed.
  - An attempt to set an environment variable failed.
- *The total data output to the standard output and standard error output exceeds 100KB* was added to the description of return value 72 of a content plug-in.
- The name of a basic plug-in was changed from File-Forwarding Plug-in to File-Transfer Plug-in.
- The name of a basic plug-in was changed from Judge ReturnCode Plug-in to Branch by ReturnCode Plug-in.
- The name of a basic plug-in was changed from Judge Value Plug-in to Branch by Property Value Plug-in.

## (6) Changes in version 10-52

### (a) Changes in the manual (3021-3-087-40)

- Linux was added as a supported operating system.
- A procedure for changing a plug-in version was added as part of the addition of the plug-in version management function.
- A description of the local execution function was added. This function allows users to start processes directly on local hosts and perform tasks such as executing commands and copying files.
- Keyboard interactive authentication was added as an authentication method used for SSH connections with operation target devices.
- A description of characters that can be specified in the **Command line** text box was added.

## (7) Changes in version 10-50

### (a) Changes in the manual (3021-3-087-30)

- A description of the case in which the user wants to use a content plug-in provided by JP1/AO in service template development was added.
- Public key authentication was added as an authentication method for operation target devices.
- An explanation that user profiles are not inherited when the OS of the operation target device is Windows was added.
- A description of the specification in the **Command line** text box to execute a non-standard script was added.

### (b) Changes in the manual (3021-3-363-10(E))

- For the manual issued in December 2014 or later, the title and reference number were changed as shown below.

Before the change:

*Job Management Partner 1/Automatic Operation GUI and Command Reference* (3021-3-315(E))

After the change:

*Job Management Partner 1/Automatic Operation GUI, Command, and API Reference* (3021-3-366(E))

- Windows Server 2012 R2 was added as a supported operating system.
- Functionality was added to facilitate the debugging of service templates. The manual structure was changed to accommodate this new functionality.
- A description of the case in which the user wants to use a content plug-in provided by JP1/AO in service template development was added.

- Public key authentication was added as an authentication method for operation target devices.
- Release plug-ins can now be deleted.
- A description of reserved properties was added. Also, the following reserved properties were added:
  - reserved.loop.index
  - reserved.service.category
  - reserved.service.name
  - reserved.service.resourceGroupName
  - reserved.step.path
  - reserved.step.prevReturnCode
  - reserved.task.description
  - reserved.task.id
  - reserved.task.name
  - reserved.task.submitter
  - reserved.task.url
- For the reserved.step.prevReturnCode reserved property, a description of a scenario in which the preceding step is not executed when retrying a task was added.
- Subsequent-step execution conditions were added as information inherited when pasting a step or relational line.
- In addition to Windows and Linux, content plug-ins that execute commands and scripts in AIX, HP-UX, and Solaris are now supported.
- The description of the execution user for commands and scripts by using content plug-ins was clarified.
- An explanation that user profiles are not inherited when the OS of the operation target device is Windows was added.
- A function was added to allow users to select whether to elevate user permission to root privilege when executing a content plug-in. This function can be used when the OS of the operation target device is UNIX.
- A description was added regarding the conditions under which files can be transferred.
- The folder in which transferred files are stored can now be set in the property file (config\_user.properties).
- The manual now mentions that certain commands must be installed in the OS of the operation target device before executing content plug-ins.
- The ibm-943 character set used for communication by JP1/AO during plug-in execution was changed to ibm-943C.
- The manual now mentions that certain commands must be installed in the OS of the operation target device before executing plug-ins.
- A description of the setting in the plugin.suPassword reserved plug-in property when execution with root privileges is not specified was added.
- A description of the specification in the **Command line** text box to execute a non-standard script was added.
- A cautionary note about specifying the command line for a content plug-in was added.
- A description of the return values of content plug-ins was added.
- A description of the relationship among the return values of executed commands or scripts, plug-ins, and steps was added.
- A procedure for using a return value as the branching condition for a flow was added, for situations when a command or script executed as a plug-in returns a value outside the 0 to 63 range.
- A section on the standard output of plug-ins was added.



- The manual now instructs users not to enclose the path of the execution directory in double or single-quotation marks, even if the path contains spaces.

## **(8) Changes in version 10-12**

### **(a) Changes in the manual (3021-3-087-20)**

- Windows Server 2012 R2 was added as a supported operating system.
- Functionality was added to facilitate the debugging of service templates. The manual structure was changed to accommodate this new functionality.
- For the reserved.step.prevReturnCode reserved property, a description of a scenario in which the preceding step is not executed when retrying a task was added.
- Subsequent-step execution conditions were added to the information inherited when pasting a step or relational line.
- A function was added to allow users to select whether to elevate user permission to root privilege when executing a content plug-in. This function can be used when the OS of the operation target device is UNIX.
- A description was added regarding the conditions under which files can be transferred.
- The folder in which transferred files are stored can now be set in the property file (config\_user.properties).
- A description of the setting in the plugin.suPassword reserved plug-in property when execution with root privileges is not specified was added.
- A description of the return values of content plug-ins was added.
- A description of the relationship among the return values of executed commands or scripts, plug-ins, and steps was added.
- A procedure for using a return value as the branching condition for a flow was added, for situations when a command or script executed as a plug-in returns a value outside the 0 to 63 range.

## **(9) Changes in version 10-11**

### **(a) Changes in the manual (3021-3-087-10)**

- Release plug-ins can now be deleted.
- A description of reserved properties was added. Also, the following reserved properties were added:
  - reserved.loop.index
  - reserved.service.category
  - reserved.service.name
  - reserved.service.resourceGroupName
  - reserved.step.path
  - reserved.step.prevReturnCode
  - reserved.task.description
  - reserved.task.id
  - reserved.task.name
  - reserved.task.submitter
  - reserved.task.url



- In addition to Windows and Linux, content plug-ins that execute commands and scripts in AIX, HP-UX, and Solaris are now supported.
- The description of the execution user for commands and scripts by using content plug-ins was clarified.
- The manual now mentions that certain commands must be installed in the OS of the operation target device before executing content plug-ins.
- The ibm-943 character set used for communication by JP1/AO during plug-in execution was changed to ibm-943C.
- The manual now mentions that certain commands must be installed in the OS of the operation target device before executing plug-ins.
- A cautionary note about specifying the command line for a content plug-in was added.
- A section on the standard output of plug-ins was added.
- The manual now instructs users not to enclose the path of the execution directory in double or single-quotation marks, even if the path contains spaces.

# Index

## A

- ability to display window items by service template configuration type and user role [228](#)
- active environments
  - reason for maintaining separate from development environment [134](#)
- adding
  - service properties [107](#), [108](#)
- auto-completion
  - property values [76](#)

## B

- basic plug-in [137](#)
- batch-updating
  - components used as steps to latest versions [98](#)
- build and release operations
  - configuration types of service templates and plug-ins assigned by [227](#)
  - reference information [225](#)
- build and release processes
  - structure and differences [225](#)
- build or release operations
  - performed on multiple instances of same service template or plug-in [227](#)
- building
  - overview [184](#)
  - service templates [188](#)

## C

- changing
  - step property values or return value during debugging [207](#)
- changing version
  - component used as step [100](#)
- character set
  - setting specific character set at plug-in execution [143](#)
  - used during plug-in execution [142](#)
- checking
  - details about debug tasks from Task Details window [219](#)
  - progress of debug task from Tasks window [218](#)
  - property mapping settings during debugging [205](#)
  - task log entries for debug tasks [219](#)
- checking versions
  - components used as steps [98](#)

- command and script return values
  - relationship to return values of plug-ins and steps [168](#)
- command or script return values
  - using as flow branching conditions (for values outside 0 to 63 range) [167](#)
- commands
  - procedure for setting [160](#)
  - required for plug-in execution [141](#)
  - specifying in CLI Command text box [165](#)
- compatibility
  - steps in service templates created in earlier versions of JP1/AO [230](#)
  - service templates [229](#)
  - service templates and plug-ins created in Service Builder window [229](#)
- component [16](#)
  - changing version to any specified version [100](#)
- component icons
  - image files that can be set for [150](#)
- component used as step
  - changing version [37](#)
- components
  - batch-updating to latest versions [98](#)
  - checking versions [98](#)
  - information inherited when changing versions [102](#)
  - managing versions [97](#)
  - overview of managing versions [97](#)
- Conditional expressions that use arrows to indicate a connection with subsequent steps [73](#)
- content plug-in [137](#)
- content plug-ins
  - return values [167](#)
- copying
  - service templates [125](#)
- creating
  - flow hierarchy [67](#)
- credential types
  - plug-in [151](#)
- custom file
  - format [54](#)
  - overview [51](#)
  - setting for service template [52](#)
  - switching for individual locales [53](#)

## D

- data type
  - whether properties can be mapped 86
- debug task flow
  - displaying 212
- debug tasks
  - managing 218
  - procedure for checking details from Task Details window 219
  - procedure for checking progress from Tasks window 218
  - procedure for deleting 222
  - procedure for forcibly stopping 221
  - procedure for stopping 220
- debug tasks that are waiting for response (response entry)
  - procedure for handling 210
- debugging
  - overview 185
  - service templates 190
  - service templates, example 194
  - settings used when starting 197
  - starting 195
  - without pausing between steps, procedure for 198
- debugging again without rebuilding
  - service templates 210
- definition
  - service resource file 58
- deleting
  - debug tasks 222
  - service templates 128
- deletion and archiving of service templates, tasks, and services during build and release operations 228
- development environments
  - reason for maintaining separate from active environment 134
- development plug-in 137
- development service template 16, 46
- development service templates
  - procedure for deleting 128
- display settings
  - for properties 113
- displaying
  - debug task flow 212
  - flow tree of debug task 214
  - repeated execution flow during debugging 215
  - step property values and return values during debugging 208

## E

- editing
  - service properties 107
- editing platform 158
- editing plug-ins to apply to service templates
  - general procedure 35
- effect of changing values of step properties during debugging 209
- elevating
  - step properties to service properties 89
- environment variables
  - procedure for adding 172
  - procedure for deleting 173
  - procedure for editing 172
- example of defining
  - step properties 90
- Execution Directory
  - specifying 171
- execution order
  - steps 75
- executors
  - plug-in 138
- exporting
  - service templates 132
  - step properties in Service Builder Debug window 208

## F

- file transfer
  - files transferred to UNIX systems 141
  - files transferred to Windows systems 140
- flow 16
  - relationship to steps 66
- flow hierarchy 67
- flow tree of debug task
  - displaying 214
- forcibly stopping
  - debug tasks 221
- format
  - service resource file 57
- functions used during debug operations 192

## G

- general procedure
  - creating new service template 31
- general procedure for debugging
  - service templates 192

general procedure for validating  
service templates 183

## H

handling  
debug tasks that are waiting for response (response entry) 210

## I

image files  
that can be set for component icons 150

importing  
service templates 133  
service templates that contain steps using service components 134  
step properties in Service Builder Debug window 208

information displayed for repeated execution plug-ins and repeated execution flows during debugging 217

information inherited  
when versions of components are changed 102

input properties of services  
items set for 109

input property values  
directly specifying 81

interrupt operation  
notes in Service Builder window 26

interrupting  
step execution during debugging 199

items  
set for variables 112  
set for input properties of services 109  
set for output properties of services 111

items to set  
plug-in definition information 149

items to set for platform 159

## M

managing  
debug tasks 218  
service templates 123

mapping  
step property values 83

## N

notes  
service share properties 117

when interrupt operation is performed in Service Builder window 26

## O

operation test  
overview 186

operations  
that can be performed on relational lines 77  
that can be performed on steps 77

Output Filter  
specifying 171

output properties of services  
items set for 111

overview  
building 184  
debugging 185  
operation test 186  
property mapping 85  
service property 107  
Service share properties 116  
service template development 14  
service template validation 183  
subsequent step conditions 72

## P

plug-in  
available operations by plug-in type 138  
credential types 151  
executors 138  
setting display information 174

plug-in definition information  
items to set 149  
procedure for editing 147

plug-in execution  
locale settings for operation target devices 141

plug-in input properties  
items to set 154

plug-in output properties  
items to set 155

plug-in processing  
procedure for skipping during debugging 203

plug-in properties  
overview 152  
procedure for adding 152  
procedure for deleting 156  
procedure for editing 153

- setting 152
  - plug-in resource file
    - automatically generated when plug-ins are created 176
    - correspondence between properties and displayed information 176
    - displaying plug-ins 177
  - plug-in resource files
    - format 175
    - procedure for setting 174
    - updated when plug-ins are edited 177
  - plug-ins
    - copying 179
    - creating 136
    - creating and adding to service templates 36
    - creating definition information 145
    - deleting 181
    - displaying by using web browser with locale for which plug-in resource file has not been created 177
    - editing 136
    - editing definition information 145
    - managing 178
    - overview 137
    - procedure for copying 179
    - procedure for creating 145
    - procedure for deleting 181
    - procedure for editing definition information 147
    - standard output 169
    - uniqueness 126
  - plug-ins that cannot be interrupted or skipped during debugging 204
  - procedure
    - adding plug-in properties 152
    - copying plug-ins 179
    - deleting plug-in properties 156
    - deleting plug-ins 181
    - editing plug-in properties 153
    - setting commands 160
    - setting plug-in resource files 174
    - skipping plug-in processing during debugging 203
  - procedure for adding
    - service share properties 117
  - procedure for deleting
    - development service templates 128
    - property groups 122
    - service properties 115
  - procedure for setting
    - property groups 121
  - procedure for copying
    - service templates 125
  - procedure for editing
    - service properties 107
  - procedure for editing platform 158
  - procedure for exporting
    - service templates 132
  - procedure for importing
    - service templates 133
  - procedure for releasing
    - service template 130
  - procedure for setting scripts
    - when attaching created scripts 163
    - when directly entering scripts 164
  - procedure for viewing
    - service templates 124
  - properties
    - display settings 113
    - visibility 113
    - whether to be mapped depending on visibility or data type 86
  - property groups
    - procedure for deleting 122
    - procedure for setting 121
    - setting 121
  - property mapping
    - overview 85
  - property mapping settings
    - procedure for checking during debugging 205
  - Property tab of Service Builder Edit window 105
  - property values
    - auto-completion 76
    - setting dynamically or statically 39
- ## R
- reference information 225
    - build and release operations 225
  - relational lines
    - behavior when connected to multiple steps 78
    - information inherited when pasting 77
    - operations that can be performed 77
    - plug-ins when processing branches 79
    - restrictions 78
  - release
    - overview 129
  - release plug-in 137
  - release service template 16, 46

- releasing
    - service templates 129
  - repeated execution flow during debugging
    - displaying 215
  - reserved plug-in properties
    - specifying authentication information 155
    - specifying execution-target hosts 155
  - reserved property
    - list 92
  - resource file
    - setting display information for service template 56
  - resource files
    - setting plug-in display information 174
  - retrying
    - task from failed step during debugging 211
    - task from step after failed step during debugging 212
  - return values
    - of content plug-ins 167
- ## S
- script specification method 161
  - scripts
    - method for specifying 161
    - procedure for setting 163, 164
  - Service Builder Debug window 190
  - Service Builder Edit window General tab 47
  - Service Builder Edit window Flow tab 64
  - Service Builder window 21
    - notes when interrupt operation is performed 26
  - service component 137
  - service properties
    - adding 107, 108
    - editing 107
    - general procedure for setting 41
    - procedure for deleting 115
    - procedure for editing 107
    - setting 104
  - service property
    - overview 107
  - service resource file
    - automatically generated at service template creation 60
    - updated when service template is edited 61
    - correspondence between properties and displayed information 59
    - definition 58
    - displaying service template in Web browser whose locale has no service resource file 62
    - format 57
    - setting 56
  - service share properties
    - notes 117
    - procedure for adding 117
  - Service share properties
    - overview 116
  - Service Share Properties 116
  - service template 16
    - active environment 16
    - adding processing 38
    - changing definition information 48
    - creating and adding plug-ins 36
    - creating and changing definition information 47
    - creating blank template 47
    - creating new 31
    - deleting processing 38
    - development environment 16
    - editing and reusing 33
    - editing definition information 34
    - elements involved in development 16
    - flow of development 13
    - list of development features 43
    - plug-in 16
    - procedure for releasing 130
    - setting custom file 52
    - setting definition information 45
    - setting display information in resource file 56
    - starting editing 25
    - tasks 28
    - tasks performed when creating 28
    - tasks performed when editing and reusing 29
    - tasks performed when using existing service template as is 29
    - transition of windows when developing 19
    - using existing template 42
    - windows used for development 19
  - service template definition information
    - items to set 49
  - service template development
    - flow 14
  - service template flow
    - creating 63
    - editing 63

- service template validation
  - overview [183](#)
- service templates
  - building [188](#)
  - compatibility for [229](#)
  - copying [125](#)
  - debugging [190](#)
  - deleting [128](#)
  - example of debugging [194](#)
  - exporting [132](#)
  - general procedure for debugging [192](#)
  - general procedure for validating [183](#)
  - importing [133](#)
  - managing [123](#)
  - procedure for building [188](#)
  - procedure for copying [125](#)
  - procedure for debugging again without rebuilding [210](#)
  - procedure for exporting [132](#)
  - procedure for importing [133](#)
  - procedure for testing operation of [223](#)
  - procedure for viewing [124](#)
  - releasing [129](#)
  - testing operation of [223](#)
  - uniqueness [126](#)
  - validating [182](#)
  - viewing [124](#)
- service templates that contain steps using service components
  - importing [134](#)
- setting
  - property groups [121](#)
- setting definition information
  - service template [45](#)
- settings
  - step definition information [72](#)
- shared built-in service properties
  - overview [118](#)
- standard error output
  - mapping to output properties [169](#)
- standard output
  - mapping to output properties [169](#)
  - plug-ins [169](#)
- starting
  - debugging [195](#)

- step [16](#)
  - changing version of component to any specified version [100](#)
  - definition information [72](#)
- step execution
  - interrupting during debugging [199](#)
- step information that can be changed when step execution is interrupted [202](#)
- step properties
  - elevating to service properties [89](#)
  - example of defining [90](#)
  - importing and exporting in Service Builder Debug window [208](#)
  - overview [81](#)
  - setting [81](#)
- step property values
  - mapping [83](#)
- step property values and return values
  - displaying during debugging [208](#)
- step property values or return value
  - procedure for changing during debugging [207](#)
- steps
  - adding [69](#)
  - adding procedure [69](#)
  - batch-updating components to latest versions [98](#)
  - checking versions of components used as [98](#)
  - defining execution order [75](#)
  - directly specifying input property values [81](#)
  - editing [69](#)
  - editing procedure [71](#)
  - information inherited when pasting [77](#)
  - managing versions of components used as [97](#)
  - operations that can be performed [77](#)
  - overview of managing versions of components used as [97](#)
  - procedure for defining execution order [75](#)
  - relationship to flow [66](#)
  - warning icon [95](#)
- stopping
  - debug tasks [220](#)
- subsequent step conditions
  - overview [72](#)

## T

- task
  - procedure for retrying from failed step during debugging [211](#)

- procedure for retrying from step after failed step during debugging [212](#)

- task log entries for debug tasks

- procedure for checking [219](#)

- testing

- operation of service template [223](#)

- operation of service templates [223](#)

- timing with which step execution can be interrupted [198](#)

## V

- validating

- service templates [182](#)

- variables

- items set for [112](#)

- viewing

- service templates [124](#)

- visibility

- for properties [113](#)

- whether properties can be mapped [86](#)

## W

- warning icon

- steps [95](#)





6-6, Marunouchi 1-chome, Chiyoda-ku, Tokyo, 100-8280 Japan

---