

**JP1 Version 11**

**JP1/Automatic Operation Command and API  
Reference**

**3021-3-A91-30(E)**

## Notices

### ■ Relevant program products

P-2A2C-E1BL JP1/Automatic Operation 11-50 (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 11-50 (for Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016)
- P-CC2A2C-EBBL JP1/Automatic Operation - Contents 11-50 (for Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016)

P-2A2C-E3BL JP1/Automatic Operation Content Pack 11-50 (for Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016)

P-822C-E1BL JP1/Automatic Operation 11-50 (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 11-50 (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 11-50 (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 11-50 (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. Active Directory is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Citrix and XenDesktop are trademarks of Citrix Systems, Inc. and/or one or more of its subsidiaries, and may be registered in the United States Patent and Trademark Office and in 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.

Microsoft and SQL Server 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.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc., in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

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

Nov. 2017: 3021-3-A91-30(E)



## ■ Copyright

All Rights Reserved. Copyright (C) 2016, 2017, Hitachi, Ltd.

## Summary of amendments

The following table lists changes in this manual (3021-3-A91-30(E)) and product changes related to this manual.

Changes	Location
JP1/AJS3 is no longer included in JP1/AO, and therefore the <code>stopcluster</code> command is no longer required. Accordingly, descriptions of this requirement were deleted.	<a href="#">1.1</a> , <a href="#">1.5</a>
MD5withRSA was deleted from the signature algorithm that can be specified by using the <code>hcnds64ssltool</code> command with the <code>sigalg</code> option specified.	<a href="#">1.5.6</a>
The value 5 was deleted from the list of return values for the <code>hcnds64dbtrans</code> command executed with the <code>export</code> option specified.	<a href="#">1.7.4</a>

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

# Preface

This manual describes the commands and API functions of 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 manual *JP1/Automatic Operation Overview and System Design Guide*.

## ■ Intended readers

This manual is intended for:

- Users of the JP1/AO commands
- Users who use the JP1/AO API

## ■ Microsoft product name abbreviations

This manual uses the following abbreviations for Microsoft product names.

Abbreviation		Full name or meaning	
Active Directory		Microsoft(R) Active Directory	
Hyper-V		Microsoft(R) Hyper-V(R)	
Internet Explorer	Microsoft Internet Explorer	Microsoft(R) Internet Explorer(R)	
	Windows 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	

Abbreviation		Full name or meaning	
Windows	Windows Server 2012 R2	Windows Server 2012 R2 Datacenter	Microsoft(R) Windows Server(R) 2012 R2 Datacenter
		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
	Windows Server Failover Cluster		Microsoft(R) Windows Server(R) Failover Cluster

## ■ Formatting conventions used in this manual

This section describes the conventions used in this manual.

### Conventions in syntax explanations

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 - 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 installation folders

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

JP1/AO installation folder

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

Common Component installation folder

/opt/HiCommand/Base64

## ■ Screenshots included in this manual

Note that, for reasons such as product improvements, the screenshots included in this manual might be partially different from the product windows you are using.



# Contents

Notices 2

Summary of amendments 5

Preface 6

## **1 Commands 13**

1.1 List of commands 14

1.2 Notes on using the commands 16

1.3 Valid characters for arguments in a command 18

1.4 Command description format 19

1.5 Configuration-related commands 20

1.5.1 encryptpassword (creating a password file) 20

1.5.2 hcmsgs64checkauth (verifying the connection with the external authentication server) 21

1.5.3 hcmsgs64fwcancel (adding an exception to the Windows Firewall exceptions list) 24

1.5.4 hcmsgs64intg (deleting or checking authentication data) 24

1.5.5 hcmsgs64ldapuser (registering and deleting users for LDAP search) 26

1.5.6 hcmsgs64ssltool (creating a private key and self-signed certificate) 28

1.5.7 setupcluster (configuring a cluster environment) 30

1.6 Operation-related commands 33

1.6.1 deleteremoteconnection (deleting a connection destination definition) 33

1.6.2 deleteservicetemplate (deleting a service template) 35

1.6.3 hcmsgs64chgurl (updating URL information) 37

1.6.4 hcmsgs64srv (starting and stopping JP1/AO, and displaying the status of JP1/AO) 38

1.6.5 hcmsgs64unlockaccount (unlocking a user account) 41

1.6.6 importservicetemplate (importing one or more service templates) 43

1.6.7 listremoteconnections (outputting the list of connection destination definitions) 45

1.6.8 listservices (outputting the list of services or service templates) 48

1.6.9 listtasks (outputting the list of tasks and the detailed task information) 52

1.6.10 setremoteconnection (adding or updating a connection destination definition) 60

1.6.11 stoptask (stopping a task) 63

1.6.12 submittask (executing a service and re-registering the tasks in a batch) 65

1.7 Maintenance-related commands 75

1.7.1 backupsystem (backing up the JP1/AO system) 75

1.7.2 hcmsgs64dbrepair (re-creating the database) 76

1.7.3 hcmsgs64dbsrv (starting and stopping the databases) 78

1.7.4 hcmsgs64dbtrans (backing up and restoring the databases) 79

1.7.5 hcmsgs64getlogs (collecting log information) 82

1.7.6 restoresystem (restoring the JP1/AO system) 86

<b>2</b>	<b>APIs 90</b>
2.1	List of APIs 91
2.2	Specifications common to APIs 96
2.2.1	Communication protocol 96
2.2.2	Security and authentication 96
2.2.3	Input/output format 97
2.2.4	Namespace 97
2.2.5	Request format 97
2.2.6	Response format 99
2.2.7	Supported methods 99
2.2.8	Domain names and resources that can be managed by APIs 100
2.2.9	Query parameter 100
2.2.10	Request header 102
2.2.11	Using HQL standard 104
2.2.12	Domain object format 105
2.2.13	Response header 106
2.2.14	Members of resources 107
2.2.15	Members to be returned for APIs that execute JP1/AO operations 128
2.2.16	Members to be returned for API functions that acquire executable operations 129
2.2.17	Status code 131
2.2.18	Error information 131
2.3	API description format 133
2.4	Service template-related API functions 134
2.4.1	Acquisition of a list of service templates 134
2.4.2	Acquisition of information about a service template 136
2.4.3	Deletion of a service template 139
2.4.4	Acquisition of a list of operations for a service template 140
2.4.5	Acquisition of the HTML file necessary for importing a service template 142
2.4.6	Import of a service template 144
2.4.7	Acquisition of information necessary for exporting a service template 146
2.4.8	Export of a service template 147
2.4.9	Acquisition of the URL for displaying the details of a service template 149
2.4.10	Acquisition of information necessary for creating a service based on a service template 150
2.4.11	Creation of a service based on a service template 152
2.5	Service-related APIs 155
2.5.1	Acquisition of a list of services 155
2.5.2	Acquisition of service information 157
2.5.3	Editing a service 159
2.5.4	Deletion of a service 162
2.5.5	Acquisition of a list of operations for a service 163
2.5.6	Acquisition of information necessary for executing a service 166

2.5.7	Execution of a service	169
2.5.8	Acquisition of information necessary for resetting the counter for a service	171
2.5.9	Reset of the counter for a service	173
2.5.10	Acquisition of information necessary for the operation to change the status of a service to release	174
2.5.11	Change of the status of a service to release	176
2.5.12	Acquisition of information necessary for the operation to change the status of a service to maintenance	177
2.5.13	Change of the status of a service to maintenance	179
2.5.14	Acquisition of information necessary for the operation to change the status of a service to disabled	181
2.5.15	Change of the status of a service to disabled	182
2.5.16	Acquisition of the URL for the details of a service	184
2.5.17	Acquisition of information necessary for changing the version of the service template used by a service	185
2.5.18	Change of the version of the service template used by a service	187
2.6	Schedule-related APIs	190
2.6.1	Acquisition of a list of schedules	190
2.6.2	Acquisition of schedule information	193
2.6.3	Acquisition of a list of operations for a schedule	194
2.6.4	Acquisition of information necessary for canceling a schedule	196
2.6.5	Cancellation of a schedule	198
2.6.6	Acquisition of information necessary for pausing a schedule	200
2.6.7	Pause of a schedule	201
2.6.8	Acquisition of information necessary for resuming a schedule	203
2.6.9	Resume of a schedule	205
2.7	Task-related APIs	208
2.7.1	Acquisition of a list of tasks	208
2.7.2	Acquisition of task information	210
2.7.3	Editing a task	212
2.7.4	Deletion of a task	215
2.7.5	Acquisition of a list of task operations	216
2.7.6	Acquisition of information necessary for stopping task execution	219
2.7.7	Stoppage of task execution	220
2.7.8	Acquisition of information necessary for forcibly stopping a task	222
2.7.9	Forced stoppage of a task	224
2.7.10	Acquisition of information necessary for re-executing a task	226
2.7.11	Re-execution of a task	228
2.7.12	Acquisition of information necessary for responding to a task that is in the status Waiting for Response	230
2.7.13	Response to a task that is in the status Waiting for Response	232
2.7.14	Acquisition of information necessary for retrying a task (retry from the failed step)	234
2.7.15	Retry from the failed step	236
2.7.16	Acquisition of information necessary for retrying a task (retry from the step after the failed step)	238

2.7.17	Retry from the step after the failed step	239
2.7.18	Acquisition of information necessary for archiving a task	241
2.7.19	Archiving a task	243
2.7.20	Acquisition of a list of steps	245
2.7.21	Acquisition of task logs	247
2.8	List of history-related API functions	253
2.8.1	Acquisition of a list of history records	253
2.8.2	Deletion of history records (with conditions specified)	255
2.8.3	Acquisition of a history record	257
2.8.4	Deletion of history records (with an ID specified)	259
2.8.5	Acquisition of a list of operations for a history record	260
2.9	Property-related APIs	262
2.9.1	Acquisition of a list of property definitions	262
2.9.2	Acquisition of property definition information	269
2.9.3	Acquisition of a list of operations for a property definition	271
2.9.4	Acquisition of lists of property definitions and property values	272
2.9.5	Acquisition of a list of property values	275
2.9.6	Batch update of property values	278
2.9.7	Acquisition of a property value	282
2.9.8	Update of a property value	283
2.9.9	Acquisition of a list of operations for a property value	285
2.9.10	Acquisition of a list of property groups	287
2.10	Service group-related API functions	290
2.10.1	Acquisition of a list of service groups	290
2.10.2	Acquisition of information about a service group	291
2.10.3	Acquisition of a list of operations for a service group	293
2.11	Tag-related API functions	295
2.11.1	Acquisition of a list of tag groups	295
2.11.2	Acquisition of a list of tags	296
2.12	API functions for information management	300
2.12.1	Acquisition of user information	300
2.12.2	Acquisition of version information	301
2.13	API usage example	303

## Appendix 308

A	Reference Information	309
A.1	Version changes	309

## Index 324

# 1

## Commands

This chapter describes the commands available in JP1/AO.

## 1.1 List of commands

The following tables list the commands available in JP1/AO.

Table 1-1: Configuration-related commands

Command name	Function	See:
<code>encryptpassword</code> (creating a password file)	Creates a password file that you can specify as an argument in a command.	1.5.1 <code>encryptpassword</code> (creating a password file)
<code>hcms64checkauth</code> (verifying the connection with the external authentication server)	Verifies the settings in the configuration file for external authentication server linkage and the connection with an external authentication server when JP1/AO links with the external authentication server.	1.5.2 <code>hcms64checkauth</code> (verifying the connection with the external authentication server)
<code>hcms64fwcancel</code> (adding an exception to the Windows Firewall exceptions list)	Adds an exception so that Windows Firewall does not block communication between the JP1/AO server and a Web browser. You use this command to change the port number on the JP1/AO server to which the Web browser connects.	1.5.3 <code>hcms64fwcancel</code> (adding an exception to the Windows Firewall exceptions list)
<code>hcms64intg</code> (deleting or checking authentication data)	Deletes authentication data stored in the repository on the server that manages user accounts. This command can also display the address of the server that stores authentication data. You use this command to delete authentication data if you failed to delete those data during the uninstallation of JP1/AO.	1.5.4 <code>hcms64intg</code> (deleting or checking authentication data)
<code>hcms64ldapuser</code> (registering and deleting users for LDAP search)	Registers the user information required for Active Directory registration information search when JP1/AO links with Active Directory. This command can also be used to delete registered user information.	1.5.5 <code>hcms64ldapuser</code> (registering and deleting users for LDAP search)
<code>hcms64ssltool</code> (creating private key and self-signed certificate)	Creates the private key, CSR, self-signed certificate, and self-signed certificate content file required for SSL connection.	1.5.6 <code>hcms64ssltool</code> (creating a private key and self-signed certificate)
<code>setupcluster</code> (configuring a cluster environment)	Configures a JP1/AO cluster environment.	1.5.7 <code>setupcluster</code> (configuring a cluster environment)

Table 1-2: Operation-related commands

Command name	Description	See:
<code>deleteremoteconnection</code> (deleting a connection destination definition)	Deletes a connection destination definition stored in JP1/AO.	1.6.1 <code>deleteremoteconnection</code> (deleting a connection destination definition)
<code>deleteservicetemplate</code> (deleting a service template)	Deletes a service template stored in JP1/AO.	1.6.2 <code>deleteservicetemplate</code> (deleting a service template)
<code>hcms64chgurl</code> (updating URL information)	Updates access (URL) information that is stored in the repository for Common Component and used for starting an application. You use this command if the system configuration change is made after operation of JP1/AO started.	1.6.3 <code>hcms64chgurl</code> (updating URL information)
<code>hcms64srv</code> (starting and stopping JP1/AO, and displaying the status of JP1/AO)	Starts and stops the services and databases of JP1/AO. This command can also display the status of the JP1/AO services.	1.6.4 <code>hcms64srv</code> (starting and stopping JP1/AO, and displaying the status of JP1/AO)
<code>hcms64unlockaccount</code> (unlocking a user account)	Unlocks a user account. You use this command when all the user accounts are locked and the users cannot log in to JP1/AO.	1.6.5 <code>hcms64unlockaccount</code> (unlocking a user account)

Command name	Description	See:
<code>importservicetemplate</code> (importing one or more service templates)	Adds one or more service templates to JP1/AO.	1.6.6 <a href="#">importservicetemplate</a> (importing one or more service templates)
<code>listremoteconnections</code> (outputting the list of connection destination definitions)	Outputs the CSV-formatted list of the connection destination definitions registered in JP1/AO.	1.6.7 <a href="#">listremoteconnections</a> (outputting the list of connection destination definitions)
<code>listservices</code> (outputting the list of services or service templates)	Outputs the CSV-formatted list of the services or service templates registered in JP1/AO.	1.6.8 <a href="#">listservices</a> (outputting the list of services or service templates)
<code>listtasks</code> (outputting the list of tasks and the detailed task information)	Outputs the CSV-formatted list of the tasks or histories. Outputs the detailed task information to a specified folder.	1.6.9 <a href="#">listtasks</a> (outputting the list of tasks and the detailed task information)
<code>setremoteconnection</code> (adding or updating a connection destination definition)	Adds or updates a connection destination definition by using a connection destination definition information file (in CSV format).	1.6.10 <a href="#">setremoteconnection</a> (adding or updating a connection destination definition)
<code>stoptask</code> (stopping a task)	Stops execution of a task by specifying the ID of the task.	1.6.11 <a href="#">stoptask</a> (stopping a task)
<code>submittask</code> (executing a service and re-registering the tasks in a batch)	Performs a service by specifying the name of the service to be performed and the property values. Re-registers the scheduled tasks and recurring tasks in a batch, based on the detailed task information output by the <code>listtasks</code> command.	1.6.12 <a href="#">submittask</a> (executing a service and re-registering the tasks in a batch)

Table 1-3: Maintenance related commands

Command name	Description	See:
<code>backupsystem</code> (backing up the JP1/AO system)	Backs up the configuration and database information of JP1/AO to store the data in the specified folder.	1.7.1 <a href="#">backupsystem</a> (backing up the JP1/AO system)
<code>hcnds64dbrepair</code> (re-creating the databases)	Forces the databases to be deleted, re-creates them, and then recovers them using the backup data. You use this command if any of the databases is corrupted, and if using the <code>restoresystem</code> command and the <code>hcnds64dbtrans</code> command with the <code>import</code> option specified cannot restore the database.	1.7.2 <a href="#">hcnds64dbrepair</a> (re-creating the database)
<code>hcnds64dbsrv</code> (starting and stopping the databases)	Starts and stops the databases of JP1/AO. You use this command when maintaining the databases.	1.7.3 <a href="#">hcnds64dbsrv</a> (starting and stopping the databases)
<code>hcnds64dbtrans</code> (backing up and restoring the databases)	Backs up and restores the databases of JP1/AO. You use this command when re-organizing the databases of JP1/AO.	1.7.4 <a href="#">hcnds64dbtrans</a> (backing up and restoring the databases)
<code>hcnds64getlogs</code> (collecting log information)	Collects log information recorded during JP1/AO operation to output the information to the archive file.	1.7.5 <a href="#">hcnds64getlogs</a> (collecting log information)
<code>restoresystem</code> (restoring the JP1/AO system)	Restores the backup data, such as the configuration and database information of JP1/AO, obtained by the <code>backupsystem</code> command.	1.7.6 <a href="#">restoresystem</a> (restoring the JP1/AO system)

## 1.2 Notes on using the commands

---

This section provides a list of notes when you use the commands.

- You must open a command prompt as an administrator if you want to execute any command that requires Administrator permissions on a Windows Server 2008 host. You can open a command prompt as an administrator by right-clicking **Command Prompt** in the **Start** menu of Windows and then selecting **Run as administrator**. However, if the User Account Control (UAC) feature is disabled, you do not have to open a command prompt as the administrator.
- When the JP1/AO server OS is Windows, if you enable **QuickEdit Mode** in a command prompt and then click the command prompt window, the window output is suspended until you disable the QuickEdit mode. For this reason, we recommend that you do not use the QuickEdit mode.
- If you want to use a command in a cluster environment, run the command in the executing host. However, note that you can run the `hcnds64getlogs` command in the standby host.
- If you want to suspend the execution of a command, press the **Ctrl+C** keys. After the command is suspended, check the suspension message for any problem. If you want to resume the command, then execute it again.
- Do not press the **Ctrl+S** keys while a command is being executed. If you do so, the command output is stopped.
- You cannot execute any commands other than the `stoptask` and `submittask` commands with other JP1/AO commands at the same time.
- You might get a return value other than 2 (The command execution has been interrupted) depending on the type of the command when you interrupt the command immediately after its execution.
- You can execute a maximum of 10 instances respectively for the `stoptask` and `submittask` command at the same time. If you attempt to execute the 11th instance, the following message appears and the task is not executed: `KNAE03236-E No more commands can be executed at the moment. Wait until one or more currently executing commands end, and then try again.`
- When you execute the following commands, you can change the subject identification information that will be output to the audit log by changing the user-specified properties file (`config_user.properties`) settings:
  - `deleteremoteconnection` command
  - `deleteservicetemplate` command
  - `importservicetemplate` command
  - `listremoteconnections` command
  - `listservices` command
  - `listtasks` command
  - `setremoteconnection` command
  - `stoptask` command
  - `submittask` command
- When the JP1/AO server OS is Linux, if the maximum size of a core file when it is output is set to 0 in the standard setting, a core dump is not generated. To generate a core dump when an error occurs, execute the `ulimit` command to set the maximum size to unlimited before executing any command.
- If the JP1/AO server OS is Windows, the specified file path is not case sensitive. If the JP1/AO server OS is Linux, the specified file path is case sensitive.



---

## Related topics

- [Topic User-specified properties file \(config\\_user.properties\) in the JP1/Automatic Operation Configuration Guide](#)
-

## 1.3 Valid characters for arguments in a command

---

This section describes the valid characters for arguments in a command.

- You can specify arguments in a command according to the specification of the OS command prompt and shell. Therefore, if an argument contains any spaces or special characters, you must escape the argument by, for example, enclosing it in double quotation marks (").
- The following characters are available when you specify a path argument in each command:  
Half-width alphanumeric characters, `_`, `.`, `-`, (space), `(`, `)`, `#`, `@`, `:`, `\`  
However, there are no limitations on the path to be specified in the `propertyfile` option of the `submittask` command.  
Note that, if the JP1/AO server OS is Linux, `/` can also only be used to separate folders.
- If the JP1/AO server OS is Windows, `:` can only be used to separate the drive letter.
- If the JP1/AO server OS is Windows, `\` can only be used to separate folders.
- You cannot specify a path in the UNC format when specifying the path as an argument.
- You cannot use a path whose folder name is preceded or followed by a space character when specifying the path as an argument. Also, you cannot use a folder name that only has the space characters.
- You cannot use a path whose folder name is preceded or followed by a period (`.`) when specifying the path as an argument. Also, you cannot use a folder name that only has the periods.
- Unless otherwise specified, you can use 1-230 characters for the absolute path.
- Unless otherwise specified, arguments for each command are case sensitive.
- The names shown below are reserved keywords in the OS. Do not use them for file and folder names.  
`CON`, `PRN`, `AUX`, `CLOCK$`, `NUL`, `COM0`, `COM1`, `COM2`, `COM3`, `COM4`, `COM5`, `COM6`, `COM7`, `COM8`, `COM9`, `LPT0`, `LPT1`, `LPT2`, `LPT3`, `LPT4`, `LPT5`, `LPT6`, `LPT7`, `LPT8`, `LPT9`

## 1.4 Command description format

---

This section explains the format of command descriptions.

Each command description has the following information. However, some commands do not have all of the information.

### Description

Describes the functionality of the command.

### Syntax

Describes the command syntax as follows:

```
command-name [ /option [value] . . . ]
```

The combination of *option* and *value* is referred to as an *option*. The term *arguments* is also used as a generic term for *options*.

### Arguments

Describes the arguments of the command.

If the JP1/AO server OS is Linux, replace / with – when you read the description.

### Located in

Shows the directory where the command is located.

If the JP1/AO server OS is Linux, replace \ with / when you read the description.

### Execute permission

Describes user permission required to execute the command.

### Remarks

Contains what you have to be aware of when you use the command.

### Return code

Lists the return codes from the command.

For details about the messages shown when the command is executed, see the manual *JP1/Automatic Operation Messages*.

Some commands output audit logs. For details about the commands that output audit logs, actions to be audited, and IDs of the messages to be output, see the topic *Event types for which audit log data is output* in the manual *JP1/Automatic Operation Administration Guide*.

### Example

Shows sample usage of the command.

## 1.5 Configuration-related commands

---

### 1.5.1 encryptpassword (creating a password file)

#### Description

This command creates a password file that you can specify as an argument in a JP1/AO command.

You can execute the command with the user ID and password of a user registered in JP1/AO and the path to the password file to be created for that user to create an encrypted password file.

By specifying the created password file instead of the password when each command is executed, specifying the password is no longer needed.

#### Syntax

```
encryptpassword
  /user user-ID
  /password password
  /passwordfile password-file-path
```

#### Arguments

*/user user-ID*

This option specifies the user ID of a JP1/AO user for which you want to create a password file.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are half-width alphanumeric characters, !, #, \$, %, &, ', (, ), \*, +, -, ., =, @, \, ^, \_, and |.

This option is not case sensitive.

*/password password*

This option specifies the password of the user indicated by the *user* option.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are the same as those for the *user* option.

*/passwordfile password-file-path*

This option specifies the absolute or relative path to the password file to be created. An error occurs if the specified path exists.

#### Located in

In Windows:

*JP1/AO-installation-folder\bin*

In Linux:

*/opt/jp1ao/bin*

## Execute permission

Execute the command as a user with Administrator or root permissions. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

Execute the command as a user with Administrator permissions. If a user without Administrator permissions executes the command, a message appears asking the user to elevate the permission level.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
4	An exclusive error occurred.
5	Communication failed.
6	Authentication failed. (The specified value is invalid.)
7	An invalid path is specified.
8	The output path already exists.
9	The specified path does not exist.
10	The specified path is not accessible.
14	You do not have permission to execute the command.
200	Creating the password file failed.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following example shows how to use the command to create, in Windows, a password file for the specified user:

```
encryptpassword /user user01 /password pass01 /passwordfile passfile
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- 

## 1.5.2 hcmds64checkauth (verifying the connection with the external authentication server)

### Description

This command verifies the settings in the configuration file for external authentication server linkage and the connection with an external authentication server when JP1/AO links with the external authentication server.

As an external authentication server, JP1/AO can link with JP1/Base or Active Directory.

This command checks whether:

- The values of the keys in the configuration file for external authentication server linkage (`exauth.properties`) that are commonly used when JP1/AO links with an external authentication server.
- The `auth.server.type` key in the configuration file for external authentication server linkage (`exauth.properties`) has a valid value specified.

When JP1/AO links with the authentication function in JP1/Base, set the `auth.server.type` key to `jp1base`. When JP1/AO links with Active Directory, set the key to `ldap`. The key is case sensitive. If the default value for the `auth.server.type` key (that is, `internal`) is specified, an error message appears indicating the setting for using the external authentication server is not enabled.

- If JP1/AO links with the authentication function in JP1/Base, this command checks whether:
  - The same host has JP1/Base and Common Component.
  - JP1/AO supports the current version of JP1/Base.
  - Users of JP1/Base can be properly authenticated.
- If JP1/AO links with Active Directory, this command checks whether:
  - The values of the keys, used for Active Directory linkage, in the configuration file for external authentication server linkage (`exauth.properties`).
  - JP1/AO can connect to Active Directory.
  - A group search can be performed if JP1/AO can connect to Active Directory.

## Syntax

```
hcnds64checkauth
  /user user-name
  /pass password
  [/summary]
```

## Arguments

`/user user-name`

This option specifies the user name which has already been registered in the external authentication server. Note that, if JP1/AO links with the authentication function in JP1/Base, you must specify a user name that does not match the user name that has been registered in the JP1/AO.

`/pass password`

This option specifies the password for the user name which has already been registered in the external authentication server.

`/summary`

This option simplifies the confirmation message that appears when the command is executed. If this option is specified, the messages to be displayed are limited to messages indicating whether each processing phase is successful or failed, error messages, and messages indicating the results. However, if an error message similar to the message indicating the results is to appear, the former error message is omitted and only the latter resulting message is displayed.

## Located in

In Windows:

```
Common-Component-installation-folder\bin
```

In Linux:

```
/opt/HiCommand/Base64/bin
```

## Execute permission

Execute the command as a user with Administrator or root permissions.

## Return code

Return code	Description
0	The command succeeded.
1-99	This code indicates the total number of syntax errors.
100	This is the return code when the number of syntax errors exceeds 100 lines.
101-199	A connection or authentication error occurred. Unit's place: Number of connection errors Ten's place: Number of authentication errors The maximum number of each place is nine. If more than nine errors occur, each place displays nine.
247	The user ID specified in the <code>user</code> option cannot be authenticated because the user ID matches the user ID which has already been registered in JP1/AO. Specify a user ID that does not match a JP1/AO user ID.
248	JP1/Base is not installed on the same host as the one on which this command is executed.
249	The unsupported version of JP1/Base is used.
250	The command is executed on the secondary server.
252	The common item setting in the definition file is incorrect.
253	External authentication linkage is not set.
254	The argument is invalid.
255	The command terminated abnormally.

## Example

The following example shows how to use the command to verify, in Windows, the connection with the external authentication server:

```
hcnds64checkauth /user test01 /pass TTdate00 /summary
```

## Related topics

- [1.3 Valid characters for arguments in a command](#)

## 1.5.3 hcnds64fwcancel (adding an exception to the Windows Firewall exceptions list)

### Description

This command adds an exception so that Windows Firewall does not block communication between the JP1/AO server and a Web browser. You use this command when you change the port number on the JP1/AO server to which the Web browser connects from the default value.

### Syntax

```
hcnds64fwcancel
```

### Located in

*Common-Component-installation-folder\bin*

### Execute permission

Execute the command as a user with Administrator permissions.

### Return code

This command has no return code. For this reason, to confirm that the processing is successful, open the Windows Firewall settings to see that your exception is properly added to the exceptions list.

To check the Windows Firewall settings, in Windows **Control Panel**, open **Windows Firewall**.

## 1.5.4 hcnds64intg (deleting or checking authentication data)

### Syntax

This command deletes authentication data stored in the repository on the server that manages user accounts. This command can also display the address of the server that stores authentication data.

You use this command to delete authentication data if you failed to delete those data during the uninstallation of JP1/AO.

### Syntax

```
hcnds64intg
  {/delete /type Automation | /print | /primary }
  /user user-ID
  /pass password
```

### Arguments

`/delete`

This option causes the command to delete authentication data.



`/type Automation`

This option specifies `Automation` as the product name of the server that stores authentication data.

`/print`

This option causes the command to display the name of the program with which authentication data is registered.

`/primary`

This option causes the command to display the host name or IP address of the server that stores authentication data.

`/user user-ID`

This option specifies the user ID for connecting the server that stores authentication data. You specify the user ID of the account with User Management permission.

`/pass password`

This option specifies the password of the account with User Management permission.

## Located in

In Windows:

`Common-Component-installation-folder\bin`

In Linux:

`/opt/HiCommand/Base64/bin`

## Execute permission

Execute the command as a user with Administrator or root permissions.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	Authentication data has already been deleted.
2	Authentication data is stored on the server on which the command is executed.
3	Authentication data is not stored on the server on which the command is executed.
4	Authentication data is not stored on the server on which the command is executed. Also, an authentication error occurred on the server that stores authentication data.
253	An authentication error occurred on the server that stores authentication data.
254	Communication with the server that stores authentication data failed.
255	The command terminated abnormally.

## Example

The following example shows how to use the command to delete, in Windows, authentication data from the server that manages user accounts:

```
hcnds64intg /delete /type Automation /user user1 /pass pass1
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- 

## 1.5.5 hcmds64ldapuser (registering and deleting users for LDAP search)

### Description

This command registers the user information required for Active Directory registration information search when JP1/AO links with Active Directory. This command can also be used to delete registered user information.

After you use this command to register the user information, execute the `hcmds64checkauth` command to verify that the information can be properly authenticated.

### Syntax

```
hcmds64ldapuser
  {/set /dn user-identifier /pass password | /delete}
  /name {server-identifier | domain-name}
  | /list
```

### Arguments

`/set`

This option causes the command to register the user information.

`/dn user-identifier`

This option specifies the user identifier of the user to be registered. Follow RFC 4514 for the possible characters.

The characters `&`, `|`, `^`, `(`, `)`, `<`, and `>` must be enclosed by double quotation marks (") or escaped with a caret (^).

If you want to specify a value that ends with `\`, escape it with `\\`.

`/pass password`

This option specifies the password for the user that is specified with the `dn` option.

`/delete`

This option causes the command to delete the registered user information. The information of the user which includes the server identifier or domain name specified by the `name` option is deleted.

`/name {server-identifier | domain-name}`

When registering the user information, specify the server identifier or domain name to which the user is registered.

When deleting the user information, specify the server identifier or domain name of the server in which the user to be deleted is registered.

However, you cannot specify the domain name if group linkage with Active Directory is disabled and a user for LDAP search is registered. In that case, specify the server identifier.

`/list`

This option causes the command to display the list of server identifiers and domain names contained in the registered user information.

## Located in

In Windows:

```
Common-Component-installation-folder\bin
```

In Linux:

```
/opt/HiCommand/Base64/bin
```

## Execute permission

Execute the command as a user with Administrator or root permissions. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The argument includes a character that cannot be specified.
3	The registered information cannot be found.
255	The command execution has been interrupted due to an error other than the above.

## Example

- Registering the user information in Windows:

To register the user information with the user name smith, belonging in the group Users, in the server with the domain name example.com, with the password qweasd00:

```
hcnds64ldapuser /set /dn "CN=suzuki,CN=Users,DC=Example,DC=com" /pass  
qweasd00 /name example.com
```

- Deleting the user information in Windows:

To delete the user information with the server name chicago:

```
hcnds64ldapuser /delete /name tokyo
```

- To display the list of registered server identifiers and domain names:

```
hcdms64ldapuser /list
```

## Output example

The following shows an example of when the list of registered server identifiers and domain names are output:

```
[ServerName]  
chicago  
washington  
newyork
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- 

## 1.5.6 hcnds64ssltool (creating a private key and self-signed certificate)

### Description

This command creates a private key, CSR, self-signed certificate, and the self-signed certificate content file that are required for SSL connection. The created files are used for the following purposes:

- The CSR is submitted to CA to obtain the SSL server certificate. You can build an SSL connection environment by combining the obtained SSL server certificate with the private key.
- You can build an SSL connection environment by combining the self-signed certificate and the private key. However, we recommend that you use this environment for test purposes because the security level is low.
- You can check the information registered in the self-signed certificate by viewing the self-signed certificate content file.

### Syntax

```
hcnds64ssltool
  /key private-key-file-name
  /csr CSR-file-name
  /cert self-signed-certificate-file-name
  /certtext self-signed-certificate-content-file-name
  [/validity self-signed certificate-expiry-date /dname identification-name-(DN) /
  sigalg signing-algorithm]
```

### Arguments

*/key private-key-file-name*

This option specifies the absolute path to the folder that stores the private key. The absolute path must include the file name of the private key.

*/csr CSR-file-name*

This option specifies the absolute path to the folder that stores the CSR. The absolute path must include the file name of the CSR.

*/cert self-signed-certificate-file-name*

This option specifies the absolute path to the folder that stores the self-signed certificate. The absolute path must include the file name of the self-signed certificate.

*/certtext self-signed-certificate-content-file-name*

This option causes the command to output the content of the self-signed certificate in the text format. Specify the absolute path to the folder that stores the file. The absolute path must include the name of the text file.

*/validity self-signed-certificate-expiry-date*

This option specifies the expiry date of the self-signed certificate in the number of days. If this option is omitted, the expiry date becomes 3,650 days. A specifiable value is a number of days until December 31, 9999.

`/dnname identification-name-(DN)`

This option specifies identification name (DN) written in the SSL server certificate in the *attribute-type=attribute-value* format. You can specify a value with multiple attribute types by separating with a comma (,). The *attribute-type* is case insensitive. The *attribute-value* cannot include a double quotation mark (") or backslash (\).

Follow RFC 2253 for character escapes.

Escape the following characters with a backslash (\).

- +, ; <=>
- A space at the top of the character string
- A space at the end of the character string
- A hash mark (#) at the top of the character string

If you omit this option, you will input the attribute values by response input according to the prompt displayed when you execute the command.

The following table describes attribute types that can be specified in this option.

Table 1-4: List of attribute types that can be specified in the identification name (DN)

Attribute type	Description of the attribute type	Prompt displayed for response input	Attribute value
CN	Common Name	Server Name	Identification name of the JP1/AO server such as a host name, IP address, and domain name <sup>#</sup>
OU	Organizational Unit Name	Organizational Unit	Organization name of a small unit such as a department or division name
O	Organization Name	Organization Name	Organization name of the company or organization <sup>#</sup>
L	Locality Name	City or Locality	Name of the city or locality (town name in Japan)
ST	State or Province Name	State or Province	Name of the state or province (prefecture in Japan)
C	Country Name	two-character country-code	Country code (JP in Japan)

#

This item is required when you use a response input.

The following shows an example of a response input.

```
Enter Server Name [default=MyHostname]:example.com
Enter Organizational Unit:Device Manager Administration
Enter Organization Name [default=MyHostname]:HITACHI
Enter your City or Locality:Sanfrancisco
Enter your State or Province:California
Enter your two-character country-code:JP
Is CN=example.com,OU=Device Manager
Administration,O=HITACHI,L=Sanfrancisco,ST=California,C=JP correct? (y/n)
[default=n]:y
```

If you made a mistake when inputting a value, enter n at the confirmation to perform the response input again.

`/sigalg signing-algorithm`

Select one of the signing algorithms below. If this option is omitted, SHA256withRSA is assumed.

- SHA1withRSA
- SHA256withRSA

## Located in

In Windows:

```
Common-Component-installation-folder\bin
```

In Linux:

```
/opt/HiCommand/Base64/bin
```

## Execute permission

Execute the command as a user with Administrator or root permissions.

## Remarks

If the attribute type CN of the SSL server certificate does not match the host name, IP address or domain name specified as the connection target from the Web browser to the JP1/AO server, a server name mismatch warning or error occurs.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
250	Deleting the key store failed.
251	Creating the private key failed.
252	Creating the self-signed certificate failed.
253	Creating the CSR failed.
254	Creating the self-signed certificate content file failed.
255	The command terminated abnormally.

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- 

## 1.5.7 setupcluster (configuring a cluster environment)

### Description

This command configures a JP1/AO cluster environment. You need to execute the command on both executing and standby hosts.

You can execute the command with the path on the shared disk to which the databases and data are backed up to configure the cluster environment.

After the command is executed, a message appears indicating ongoing processes.

## Syntax

```
setupcluster  
  /exportpath path-to-which-the-databases-and-data-are-backed-up
```

## Arguments

*/exportpath path-to-which-the-databases-and-data-are-backed-up*

This option specifies the absolute or relative path to the folder to which the databases and data on which this command is executed are backed up. You must specify the folder on the shared disk that has sufficient free space. The maximum length of the path name is 49 characters.

## Located in

In Windows:

*JP1/AO-installation-folder\bin*

In Linux:

*/opt/jplao/bin*

## Execute permission

Execute the command as a user with Administrator or root permissions. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
4	One of the other commands is running.
5	Communication failed.
7	An invalid path is specified.
9	The specified path does not exist.
10	The specified path is not accessible.
11	The specified folder is not empty.
14	You do not have permission to execute the command.
120	Setting up the cluster failed.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following example shows how to use the command to configure a cluster environment in Windows. In this example, the databases are re-created and the data is backed up in the path on the shared folder specified on the executing host. (In the case of the standby host, the same command is used.)

```
setupcluster /exportpath Z:\share
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
-



## 1.6 Operation-related commands

---

### 1.6.1 deleteremoteconnection (deleting a connection destination definition)

#### Description

This command deletes a connection destination definition stored in JP1/AO.

You can execute the command with the ID of a connection destination definition that you want to delete to delete the specified connection destination definition.

#### Syntax

```
deleteremoteconnection
  /id ID-of-the-connection-destination-definition
  /user user-ID
  {/password password | /passwordfile password-file-path}
```

#### Arguments

`/id`

This option specifies the ID of the connection destination definition that you want to delete. Note that you need to execute the `listremoteconnections` command beforehand to check the ID of the connection destination definition.

The number of possible characters is in the range from 1 to 64 characters.

The possible characters are half-width numeric characters.

`/user`

This option specifies the user ID for JP1/AO.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are half-width alphanumeric characters, `!`, `#`, `$`, `%`, `&`, `'`, `(`, `)`, `*`, `+`, `-`, `.`, `=`, `@`, `\`, `^`, `_`, and `|`.

This option is not case sensitive.

`/password`

This option specifies the password of the user indicated by the `user` option.

You must specify either this option or the `passwordfile` option. If both options are specified, or if neither is specified, then you will get an error.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are the same as those for the `user` option.

`/passwordfile`

This option specifies the absolute or relative path to the password file for the user specified in the `user` option. You can create a password file by using the `encryptpassword` command.

You must specify either this option or the `password` option. If both options are specified, or if neither is specified, then you will get an error.

## Located in

In Windows:

```
JP1/AO-installation-folder\bin
```

In Linux:

```
/opt/jplao/bin
```

## Execute permission

Execute the command as a user with both Administrator or root permissions for the OS and with the Admin role for JP1/AO. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

## Remarks

This command can delete one connection destination definition each time the command is executed.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
4	One of the other commands is running.
5	Communication failed.
6	Authentication failed.
14	You do not have permission to execute the command.
240	Deleting the connection destination definition failed.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following example shows how to use the command to delete, in Windows, the specified connection destination definition (whose connection destination definition ID is 12345):

```
deleteremoteconnection /id 12345 /user user01 /password pass01
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
-

## 1.6.2 deleteservicetemplate (deleting a service template)

### Description

This command deletes a service template stored in JP1/AO.

You can execute the command with the service template ID, vendor ID, and version number of a service template that you want to delete to delete the specified service template.

### Syntax

```
deleteservicetemplate
  /name service-template-ID
  /vendor vendor-ID-of-the-service-template
  /version version-number-of-the-service-template
  /user user-ID
  {/password password | /passwordfile password-file-path}
```

### Arguments

*/name service-template-ID*

This option specifies the service template ID of the service template that you want to delete.

This option is not case sensitive.

The number of possible characters is in the range from 1 to 64 characters.

The possible characters are half-width alphanumeric characters, -, \_, and ..

*/vendor vendor-ID-of-the-service-template*

This option specifies the vendor ID of the service template that you want to delete.

This option is not case sensitive.

The number of possible characters is in the range from 1 to 64 characters.

The possible characters are half-width alphanumeric characters, -, \_, and ..

*/version version-number-of-the-service-template*

This option specifies the version number of the service template that you want to delete in *XX.YY.ZZ* format.

The possible characters for *XX*, *YY*, and *ZZ* are two-digit half-width numeric characters, which are from 00 through 99.

- *XX*: Major version number
- *YY*: Minor version number
- *ZZ*: Revision number

*/user user-ID*

This option specifies the user ID for JP1/AO.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are half-width alphanumeric characters, !, #, \$, %, &, ', (, ), \*, +, -, ., =, @, \, ^, \_, and |.

This option is not case sensitive.

*/password password*

This option specifies the password of the user indicated by the *user* option.

You must specify either this option or the `passwordfile` option. If both options are specified, or if neither are specified, then you will get an error.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are the same as those for the `user` option.

`/passwordfile password-file-path`

This option specifies the absolute or relative path to the password file for the user specified in the `user` option. You can create a password file by using the `encryptpassword` command.

You must specify either this option or the `password` option. If both options are specified, or if neither are specified, then you will get an error.

## Located in

In Windows:

`JP1/AO-installation-folder\bin`

In Linux:

`/opt/jplao/bin`

## Execute permission

Execute the command as a user with both Administrator or root permissions for the OS and with the Admin or Develop role for JP1/AO. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

## Remarks

Use this command to delete a release service template. Note that you need to use the **Editor** window to delete a development service template.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
4	One of the other commands is running.
5	Communication failed.
6	Authentication failed.
7	An invalid path is specified.
9	The specified path does not exist.
14	You do not have permission to execute the command.
190	Deleting the service template failed.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following example shows how to use the command to delete, in Windows, the specified service template (whose service template ID is nameA, vendor ID is vendorB, and version number is 01.00.00):

```
deleteservicetemplate /name nameA /vendor vendorB /version 01.00.00 /user
user01 /password pass01
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
  - [Topic Procedure for deleting development service templates in the JP1/Automatic Operation Service Template Developer's Guide](#)
- 

## 1.6.3 hcnds64chgurl (updating URL information)

### Description

This command updates access (URL) information that is stored in the repository for Common Component and used for starting an application.

You use this command if either of the following configuration changes is made after operation of JP1/AO started:

- If the port used by a host that has Common Components installed is changed
- If the host name or IP address of a host that has Common Components installed is changed

### Syntax

```
hcnds64chgurl
  {/list |
   /change URL-before-change URL-after-change |
   /change URL-after-change /type Automation}
```

### Arguments

*/list*

This option causes the command to display the list of URLs and product names currently set up.

*/change URL-before-change URL-after-change*

This option causes the command to overwrite the URL related information currently registered with the new URL related information.

You specify both the URL that is currently registered and the new URL. If you use the option together with the *type* option, you only specify the new URL.

If you specify a URL with an IPv6 address, enclose the IP address in [ ].

*/type Automation*

This option specifies *Automation* as the name of the product whose URL is to be changed.

## Located in

In Windows:

```
Common-Component-installation-folder\bin
```

In Linux:

```
/opt/HiCommand/Base64/bin
```

## Execute permission

Execute the command as a user with Administrator or root permissions.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The URL cannot be found.
253	Restoring the repository failed.
254	Backing up the repository failed.
255	The command terminated abnormally.

## Example

The following examples show how to use the command for each case.

- To display, in Windows, the list of URLs and product names currently set up:  

```
hcnds64chgurl /list
```
- To overwrite, in Windows, the URL related information currently registered with the new URL related information:  

```
hcnds64chgurl /change "http://192.168.11.33:22015" "http://  
192.168.11.55:22015"
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- 

## 1.6.4 hcnds64srv (starting and stopping JP1/AO, and displaying the status of JP1/AO)

### Description

This command starts and stops the services and databases of JP1/AO. This command can also display the status of the JP1/AO services or change how to start the services.

Note that if you execute this command by specifying `AutomationWebService` for the `server` option, you can start, stop, or display the status of, the services listed in the table below.

Table 1-5: List of services that can be targets of this command

Service display name and process	Starting	Stopping	Displaying status
HAutomation Engine Web Service	Y	Y	Y
HBase 64 Storage Mgmt SSO Service	Y	Y #1	Y
HBase 64 Storage Mgmt Web Service	Y	Y #1	Y
HBase 64 Storage Mgmt Web SSO Service	Y	Y #1	Y
Database process <sup>#2</sup>	Y	Y #1	Y

**Legend:**

Y: The command works. N: The command does not work.

**#1:**

The service does not stop while a service from the Hitachi Command Suite products is running.

**#2**

These are the JP1/AO internal processes. The `hcnds64srv` command does not start and stop `HiRDB/EmbeddedEdition _HD1` that represents the database service.

## Syntax

```
hcnds64srv
  {/start | /stop | /check| /status}
  [/server service-name]
```

To see the status of services from JP1/AO and all the Hitachi Command Suite products:

```
hcnds64srv
  /statusall
```

To change how to start a service or services:

```
hcnds64srv
  /starttype {auto | manual}
  {/server service-name | /all}
```

## Arguments

**/start**

This option causes the command to start the service and database specified in the `server` option.

**/stop**

This option causes the command to stop the service and database specified in the `server` option.

**/check**

This option causes the command to display the status of the service and database specified in the `server` option.

**/status**

This option causes the command to display the status of the service and database specified in the `server` option.

`/server service-name`

If you want to start and stop only the service, or display its status, of the JP1/AO product, specify `AutomationWebService` for `service-name`. If this option is omitted, the command has an effect on the services from JP1/AO and all Hitachi Command Suite products that are installed.

`/statusall`

This option causes the command to display the status of the services and databases, and of the services from the Hitachi Command Suite products that are registered with Common Component.

`/starttype {auto | manual}`

This option specifies the start type of the service specified in the `server` option.

To start the service automatically, use `auto`. To start the service manually, use `manual`.

`/all`

If this option is specified, the command has an effect on the services from JP1/AO and all Hitachi Command Suite products that are installed.

## Located in

In Windows:

`Common-Component-installation-folder\bin`

In Linux:

`/opt/HiCommand/Base64/bin`

## Execute permission

Execute the command as a user with Administrator or root permissions.

## Remarks

- When you start and stop the services for JP1/AO in day-to-day operations, start and stop all the services without specifying the `server` option. If you want to start only the services from the JP1/AO products with `server` option, use `HBase` for the `server` option to start the services from Common Component because these services must be started beforehand.
- Executing the command with the `stop` option while a task is being processed terminates any processing running on the connection destination. For this reason, if any task is in execution status (In Progress, Waiting for Response, Abnormal Detection, or Terminated), you need to wait the status transition of the task to one of the ended status (Completed, Failed, or Canceled) or stop the execution of all the tasks, and then use the command with the option.
- If the service does not stop within three minutes after the command with the `stop` option, the command terminates abnormally with a message indicating a timeout. In this case, wait a little while and then execute the command with the `stop` option again.

## Return code

The following table lists the return codes from the command with the `/start` or `stop` option.

Return code	Description
0	The command succeeded.
1	The service has already started (with the <code>start</code> option). The service has already stopped (with the <code>stop</code> option).



Return code	Description
255	The command execution failed.

The following table lists the return codes from the command with the `check`, `status`, or `statusall` option.

Return code	Description
0	The service is not running.
1	The service is running.
255	The command execution failed.

The following table lists the return codes from the command with the `starttype` option.

Return code	Description
0	The command succeeded.
255	The command execution failed.

## Example

The following examples show how to use the command for each case.

- To start, in Windows, the services from the JP1/AO products:  
`hcnds64srv /start /server AutomationWebService`
- To stop, in Windows, the services from the JP1/AO products:  
`hcnds64srv /stop /server AutomationWebService`
- To check, in Windows, the status of the services from the JP1/AO products:  
`hcnds64srv /status /server AutomationWebService`

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- 

## 1.6.5 hcnds64unlockaccount (unlocking a user account)

### Description

This command unlocks a user account. You use this command when all the user accounts are locked and the users cannot log in to JP1/AO.

### Syntax

```
hcnds64unlockaccount
  /user user-ID
  /pass password
```

## Arguments

`/user user-ID`

This option specifies the user ID of the user account that you want to unlock. You must specify the user ID with User Management permission.

`/pass password`

This option specifies the password of the user account that you want to unlock.

## Located in

In Windows:

`Common-Component-installation-folder\bin`

In Linux:

`/opt/HiCommand/Base64/bin`

## Execute permission

Execute the command as a user with Administrator or root permissions.

## Remarks

- Only a user account with User Management permission has the ability to unlock user accounts by using the `hcnds64unlockaccount` command.
- If the user name or password specified in the options includes characters, `&`, `|`, or `^`, enclose the character with double quotation marks (`"`) or escape the character with a caret (`^`). For example, in Windows, if the password is `^a^b^c^`, the command can be written as `hcnds64unlockaccount /user system /pass "^^a^^b^^c^^"` or `hcnds64unlockaccount /user system /pass ^^a^^b^^c^^`.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
251	An authentication error occurred due to login failure.
252	An authentication error occurred due to a lack of User Management permission.
253	The communication with the authentication server failed.
254	The command is executed on the secondary server.
255	The command terminated abnormally.

## Example

The following example shows how to use the command to unlock, in Windows, the specified user (whose user ID is `test01`):

```
hcnds64unlockaccount /user test01 /pass TTdate00
```

---

## Related topics

- 1.3 Valid characters for arguments in a command
- 

## 1.6.6 importservicetemplate (importing one or more service templates)

### Description

This command adds one or more service templates to JP1/AO. Adding service templates to JP1/AO is called *importing of service templates*.

You can execute the command with a single service template package or a zip file in which multiple service template packages are archived to import the specified service template package into JP1/AO.

### Syntax

```
importservicetemplate
  /file service-template-package-or-zip-file-in-which-multiple-service-template-
  packages-are-archived
  /user user-ID
  {/password password | /passwordfile password-file-path}
```

### Arguments

*/file service-template-package-or-zip-file-in-which-multiple-service-template-packages-are-archived*

This option specifies the absolute or relative path to the service template package to be imported or zip file in which multiple service template packages are archived.

*/user user-ID*

This option specifies the user ID for JP1/AO.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are half-width alphanumeric characters, !, #, \$, %, &, ', (, ), \*, +, -, ., =, @, \, ^, \_, and |.

This option is not case sensitive.

*/password password*

This option specifies the password of the user indicated by the `user` option.

You must specify either this option or the `passwordfile` option. If both options are specified, or if neither are specified, then you will get an error.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are the same as those for the `user` option.

*/passwordfile password-file-path*

This option specifies the absolute or relative path to the file that stores the password of the user specified in the `user` option.

You must specify either this option or the `password` option. If both options are specified, or if neither are specified, then you will get an error.

## Located in

In Windows:

```
JP1/AO-installation-folder\bin
```

In Linux:

```
/opt/jplao/bin
```

```
JP1/AO-installation-folder\bin
```

## Execute permission

Execute the command as a user with both the Admin or Develop role and Administrator or root permissions for the OS. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

## Remarks

When the command imports a zip file in which multiple service template packages are archived, the command continues processing even if some of the service template packages cannot be imported. Messages inform you of the service template packages that could not be imported. If such messages are displayed, check the messages, correct the causes of the failures, and then re-import the relevant service template packages.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
4	One of the other commands is running.
5	Communication failed.
6	Authentication failed.
7	An invalid path is specified.
9	The specified path does not exist.
14	You do not have permission to execute the command.
180	Importing the service template failed.
255	The command terminated abnormally.

## Example

The following examples show how to use the command for each case.

- To import, in Windows, the specified service template package (C:\temp\aaa.st) into JP1/AO:  

```
importservicetemplate /file C:\temp\aaa.st /user user1 /password pass1
```

- To import, in Windows, a zip file (C:\temp\bbb.zip) in which the specified multiple service template packages are archived into JP1/AO:

```
importservicetemplate /file C:\temp\bbb.zip /user user1 /password pass1
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
  - [Topic Notes on defining Service Share Properties in the JP1/Automatic Operation Service Template Developer's Guide](#)
- 

## 1.6.7 listremoteconnections (outputting the list of connection destination definitions)

### Description

This command outputs the CSV-formatted list of the connection destination definitions registered in JP1/AO.

### Syntax

```
listremoteconnections
  /file output-file-path
  /user user-ID
  {/password password | /passwordfilepassword-file-path}
```

### Arguments

#### /file

This option specifies the absolute or relative path to the file to which connection destination definition information is to be output. An error occurs if the specified file exists.

#### /user

This option specifies the user ID for JP1/AO.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are half-width alphanumeric characters, !, #, \$, %, &, ', (, ), \*, +, -, ., =, @, \, ^, \_, and |.

This option is not case sensitive.

#### /password

This option specifies the password of the user indicated by the `user` option.

You must specify either this option or the `passwordfile` option. If both options are specified, or if neither is specified, then you will get an error.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are the same as those for the `user` option.

#### /passwordfile

This option specifies the absolute or relative path to the password file for the user specified in the `user` option. You can create a password file by using the `encryptpassword` command.

You must specify either this option or the `password` option. If both options are specified, or if neither is specified, then you will get an error.

## Output format

The output items are output in CSV format in a single line per connection destination definition.

The values for each output item are enclosed in double quotation marks ("). Any double quotation mark (") contained in the value is escaped with another one added in front of the mark.

Table 1-6: Output format of a connection destination definition information file

Output item	Description
Id	ID of the connection destination definition
Method	One of the following values is output as the connection destination type. <ul style="list-style-type: none"><li>• IPv4</li><li>• IPv6</li><li>• HostName</li></ul>
IP Address/Host Name	IP address or host name of the host to connect to
Service Group	Service group that is assigned to the connection destination definition
Authentication	Whether authentication information has been set is output. <ul style="list-style-type: none"><li>• Enable Authentication information has been set.</li><li>• Disable Authentication information has not been set.</li></ul>
Protocol	The authentication protocol used for communication with the host to connect to is output. <ul style="list-style-type: none"><li>• Windows</li><li>• SSH</li><li>• Telnet</li></ul>
SSH Authentication Method <sup>#1</sup>	The authentication method used for communication with the host to connect to is output. <ul style="list-style-type: none"><li>• Password Authentication Password authentication</li><li>• Public Key Authentication Public key authentication</li><li>• Keyboard Interactive Authentication Keyboard interactive authentication</li></ul>
User ID	User ID for logging in to the host to connect to
Password <sup>#2</sup>	Password for logging in to the host to connect to
Superuser's Password <sup>#2</sup>	Password for the superuser of the host to connect to
Connection Status	The status when JP1/AO last connected to the host is output. <ul style="list-style-type: none"><li>• Connection Successful Connection was successful.</li><li>• Error Connection failed.</li><li>• Unknown Not connected</li><li>• - Not applicable</li></ul> If the range of the hosts to connect to is specified, a hyphen (-) will be displayed.
Connected Time	The time at which JP1/AO last connected to the host is output.

#1

If Protocol is Windows or Telnet, an empty string is output.

#2

\*\*\*\*\* is output regardless of whether the password is set.

## Located in

In Windows:

*JP1/AO-installation-folder\bin*

In Linux:

*/opt/jp1ao/bin*

## Execute permission

Execute the command as a user with both Administrator or root permissions for the OS and with the Admin role for JP1/AO. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
4	One of the other commands is running.
5	Communication failed.
6	Authentication failed.
7	An invalid path is specified.
8	The file with the same name already exists in the output path.
9	The specified path does not exist.
10	The specified path is not accessible.
13	Outputting the file failed.
14	You do not have permission to execute the command.
220	Obtaining the list of connection destination definitions failed.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following example shows how to use the command to output, in Windows, connection destination definition information as the file `C:\temp\list01.csv`:

```
listremoteconnections /file C:\temp\list01.csv /user user01 /password pass01
```

---

## Related topics

- 1.3 Valid characters for arguments in a command
- 

## 1.6.8 listservices (outputting the list of services or service templates)

### Description

This command outputs the CSV-formatted list of the services or service templates registered in JP1/AO.

You can output the list of the services, including the vendor name and version number, or of the service templates. Note that debug services are not output.

### Syntax

```
listservices
  /output {services | servicetemplates}
  /file output-file-path
  [/encoding {UTF-8 | Shift_JIS}]
  /user user-ID
  [/password password | /passwordfile password-file-path]
```

### Arguments

*/output* {services | servicetemplates}

This option specifies which information is output in the list.

- *services*  
Outputs the list of services.
- *servicetemplates*  
Outputs the list of service templates.

*/file* *output-file-path*

This option specifies the absolute or relative path to the output file. An error occurs if the specified file exists.

*/encoding* {UTF-8 | Shift\_JIS}

This option specifies the encoding of the output file. If this option is omitted, the default encoding of the OS is used.

*/user* *user-ID*

This option specifies the user ID for JP1/AO.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are half-width alphanumeric characters, !, #, \$, %, &, ', (, ), \*, +, -, ., =, @, \, ^, \_, and |.

This option is not case sensitive.

*/password* *password*

This option specifies the password of the user indicated by the *user* option.

You must specify either this option or the *passwordfile* option. If both options are specified, or if neither are specified, then you will get an error.

The number of possible characters is in the range from 1 to 256 characters.



The possible characters are the same as those for the `user` option.

`/passwordfile password-file-path`

This option specifies the absolute or relative path to the password file for the user specified in the `user` option. You can create a password file by using the `encryptpassword` command.

You must specify either this option or the `password` option. If both options are specified, or if neither are specified, then you will get an error.

## Output format

The output items are output in CSV format in a single line per service or service template.

The values for each output item are enclosed in double quotation marks ("). Any double quotation mark (") contained in the value is escaped with another one added in front of the mark.

Table 1-7: Output format of the list of services or service templates

Type of output information	Output item	Content
List of services	Name	Service name
	Favorite	Favorite-setting state
	Description	Description
	Service Group	Service group name
	Service Template	Service template name
	Vendor Name	Vendor name
	Version	Service version
	Tags	Tags set for the service
	Status	Status
	Create Date	Creation date and time
	Modify Date	Date and time of the last modification
	Submit Date	Date and time of the last submission
	Reset Date	Reset date and time
	Executed Count	Number of task executions
	Completed Count	Number of successful terminations
	Last Failed Date	Date and time of the last failure
	Failed Count	Number of failed attempts
	Submit Count	Number of service executions
	ID	Service ID
	Latest	Whether the service template is the latest version
Supported Schedule Type	Selectable schedule types	
Supported Action Type	Operations that can be performed for the task	
List of service templates	Name	Service template name
	Vendor	Vendor name

Type of output information	Output item	Content
List of service templates	Version	Service template version
	Description	Description
	Service Template Key Name	Service template ID
	Vendor ID	Vendor ID
	Tags	Tags set for the service template
	Registered	Creation time
	Updated	Time of the last update
	Latest Version	Whether the service template is the latest version
	Used Services	Number of services that use the service template
	Used Service Templates	Number of service templates that use this service template as a component
	Outdated Services	Whether any services are using an outdated version of the service template
	Outdated Component	Whether an outdated component is being used
	Supported Schedule Type	Selectable schedule types
	Supported Action Type	Operations that can be performed for tasks that use this service template
Release State	Release state	

The following list shows some examples of file outputs.

- For the list of services

```
"Name", "Favorite", "Description", "Service Group", "Service Template", "Vendor
Name", "Version", "Tags", "Status", "Create Date", "Modify Date", "Submit Date", "Reset
Date"
, "Executed Count", "Completed Count", "Last Failed Date", "Failed Count", "Submit
Count", "ID", "Latest", "Supported Schedule Type", "Supported Action Type"
"Remote Command Execution", "false", "The service executes the commands on the
remote target server.", "DefaultServiceGroup", "Remote command execution", "Hitachi,
Ltd", "01.12.00", "Basic, OS_Operations"
, "Release", "2015-08-28 13:07:25", "2015-08-28 13:07:25", "2015-08-28
13:20:26", "", "3", "1", "2015-08-28
13:17:58", "2", "3", "4005", "Yes", "immediate, schedule, recurrence", "forciblyStop, retry
"
```

- For the list of service templates

```
"Name", "Vendor", "Version", "Description", "Service Template Key Name", "Vendor
ID", "Tags", "Registered", "Updated", "Latest Version", "Used Services", "Used Service
Templates", "Outdated Services", "Outdated Component", "Supported Schedule
Type", "Release State", "Supported Action Type"
"Get List of Users from Server", "Hitachi, Ltd.", "02.00.00", "Acquires a list of
Windows or UNIX OS
users.", "osShowUsers", "com.hitachi.software.dna.cts.jp1", "AIX, Gather OS
information, Linux, Windows", "2016-11-17 13:41:21", "2016-11-17
13:41:21", "Yes", "0", "0", "No", "No", "immediate, schedule, recurrence", "Release", "forci
blyStop, retry"
```

## Located in

In Windows:

```
JP1/AO-installation-folder\bin
```

In Linux:

```
/opt/jplao/bin
```

## Execute permission

Execute the command as a user with Administrator permissions for the OS.

To output a list of services, the Admin, Develop, Modify, or Submit role must be set for the target service groups from the user group that the user who executes the command belongs to. The command does not output a list of services for any service groups for which none of these roles are set.

To output a list of service templates, the Admin, Develop, or Modify role must be set for the target service groups from the user group that the user who executes the command belongs to.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
4	One of the other commands is running.
5	Communication failed.
6	Authentication failed.
7	An invalid path is specified.
8	The file with the same name already exists in the output path.
9	The specified path does not exist.
10	The specified path is not accessible.
12	An invalid encoding is specified.
13	Outputting the file failed.
14	You do not have permission to execute the command.
160	Obtaining the list of services failed.
161	Obtaining the list of service templates failed.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following examples show how to use the command for each case.

- To output, in Windows, the list of registered services to a file in default encoding of the OS:  
`listservices /output services /file list01 /user user01 /password pass01`
- To output, in Windows, the list of registered service templates to a file in UTF-8 encoding:  
`listservices /output servicetemplates /file list02 /encoding UTF-8 /user user02 /password pass02`

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- 

## 1.6.9 listtasks (outputting the list of tasks and the detailed task information)

### Description

The functionality of the `listtasks` command is as follows:

### Outputting the list of tasks or task histories

If you execute the `listtasks` command by specifying `tasks` for the `output` option, you can output the task information displayed in the list of tasks in the **Tasks** window in a CSV-formatted file. Alternatively, if you execute the command by specifying `histories` for the `output` option, you can output the task information displayed in the list of histories in the **Tasks** window in a CSV-formatted file. Note that debug tasks are not output to either of the CSV files.

In addition, you can specify a period to filter the task information to be output.

### Outputting the detailed task information

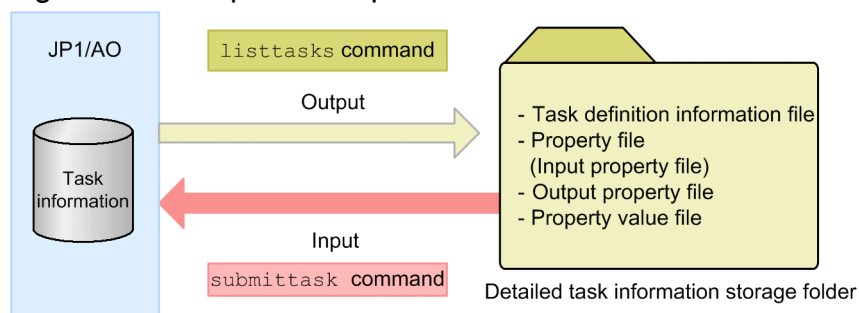
If you execute the `listtasks` command by specifying `taskdetails` for the `output` option, the detailed task information including input property and output property values is output to the detailed task information storage folder. Note that the detailed information on the debug tasks is not output.

If you execute the `submittask` command based on the detailed task information<sup>#</sup> output by the `listtasks` command, you can re-register the scheduled tasks and recurring tasks with the same setting in a batch.

#

This detailed task information does not include the definition information (service, service template, user, user group, service group, connection destinations, service share properties) and the definition file. Use the `backupsystem` command to back up those pieces of information.

Figure 1-1: Output and input of the detailed task information



## Syntax

```
listtasks
  [/startrange {yyyy-mm-dd|,yyyy-mm-dd|yyyy-mm-dd,yyyy-mm-dd}]
  /output {tasks | histories | taskdetails}
  {/fileoutput-file-path | /taskdetaildir detailed-task-information-storage-
  folder-path}
  [/encoding {UTF-8 | Shift_JIS}]
  /user user-ID
  {/password password | /passwordfile password-file-path}
```

## Arguments

`/startrange {yyyy-mm-dd|,yyyy-mm-dd|yyyy-mm-dd,yyyy-mm-dd}`

This option specifies the start date or scheduled start date of tasks to filter the tasks to be output.

`yyyy` must have a four-digit year in half-width numeric characters. `mm` must have a month from 1 (or 01) to 12 in half-width numeric characters. `dd` must have a day from 1 (or 01) to 31 in half-width numeric characters.

- `yyyy-mm-dd`  
This option causes the command to output the tasks that started or are scheduled to start on and after the specified date.
- `,yyyy-mm-dd`  
This option causes the command to output the tasks that started or are scheduled to start on and before the specified date.
- `yyyy-mm-dd,yyyy-mm-dd`  
This option causes the command to output the tasks that started or are scheduled to start within the specified period. The date on the right side of `,` can accept any date on and after the date specified on the left side.

For recurring tasks, tasks scheduled to start up to the next time are output, and any tasks scheduled to start subsequently are not output.

If you want to output waiting tasks, specify a scheduled date and time instead of a start data and time if the tasks are recurring and scheduled tasks. If they are immediate tasks, specify a submitted data and time. You can check the submitted date and time in the **Task Details** window.

If this option is omitted, all the tasks viewable to users are output.

Note that an error occurs if you specify the `startrange` option when `taskdetails` is specified for the output option.

`/output {tasks | histories | taskdetails}`

This option specifies which one of the following information is output in the list:

- `tasks`  
Outputs the list of tasks from the **Tasks** window.
- `histories`  
Outputs the list of histories from the **Tasks** window.
- `taskdetails`  
Outputs the detailed task information including the input property and output property values.

`{/file output-file-path | /taskdetaildir detailed-task-information-storage-folder-path}`

- `/file output-file-path`

This option specifies the absolute or relative path to the file in which the list is output. An error occurs if the specified file exists.

This option is required if `tasks` or `histories` is specified for the `output` option. An error occurs if this option is specified when `taskdetails` is specified for the `output` option.

- `/taskdetaildir detailed-task-information-storage-folder-path`

This option specifies the absolute or relative path to an empty folder to which the detailed task information is output. Note that only a folder on the local disk can be specified. The number of characters that can be specified is no more than 180 characters for the absolute path. If the relative path is used, the path being converted to the absolute path must be no more than 180 characters.

An error occurs if the specified folder does not exist, or the specified folder already contains a file or folder.

This option is required if `taskdetails` is specified for the `output` option. An error occurs if this option is specified when `tasks` or `histories` is specified for the `output` option.

`/encoding {UTF-8 | Shift_JIS}`

This option specifies the encoding of the output file. If this option is omitted, the default encoding of the OS is used.

If `taskdetails` is specified for the `output` option, the encoding specified here is applied only to the task list file (`listtasks.csv`) located directly under the detailed task information storage folder. The detailed task information, property file (input property file), and output property file are always output in UTF-8.

`/user user-ID`

This option specifies the user ID for JP1/AO.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are half-width alphanumeric characters, `!`, `#`, `$`, `%`, `&`, `'`, `(`, `)`, `*`, `+`, `-`, `.`, `=`, `@`, `\`, `^`, `_`, and `|`.

This option is not case sensitive.

`/password password`

This option specifies the password of the user indicated by the `user` option.

You must specify either this option or the `passwordfile` option. If both options are specified, or if neither are specified, then you will get an error.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are the same as those for the `user` option.

`/passwordfile password-file-path`

This option specifies the absolute or relative path to the password file for the user specified in the `user` option. You can create a password file by using the `encryptpassword` command.

You must specify either this option or the `password` option. If both options are specified, or if neither are specified, then you will get an error.

## Output format

The format of output from the `listtasks` command is as follows:

### When outputting the list of tasks and histories (when `tasks` or `histories` is specified for the `output` option)

The output items are output in a single line per task in CSV format.

The values for each output item are enclosed in double quotation marks (`"`). Any double quotation mark (`"`) contained in the value is escaped with another one added in front of the mark.

Table 1-8: Output items in the list of tasks

Output item	Content
Task Name	Task name
To Do	To Do setting state
Status	Status of the task
Scheduled Time	Scheduled start date and time
Start Time	Start date and time
Completion Time	End date and time
Schedule Type	Task type
Task ID	Task ID
Description	Task description
Service	Service name
Service Group	Service group
Tags	Tags set for the service
Submitted By	User who executed the task
Submit Time	Submitted date and time
Schedule Interval	Recurrence pattern
Recurrence Time	Recurrence time
Schedule Start Date	Start date of recurrence
Notes	Memo
Step Start Time	Step start date and time
Supported Action Type	Operations that can be performed for the task
Service status	Service status

#

The configuration type is output only if the Admin or Develop role has been set for the target resource groups from the user group that the user belongs to.

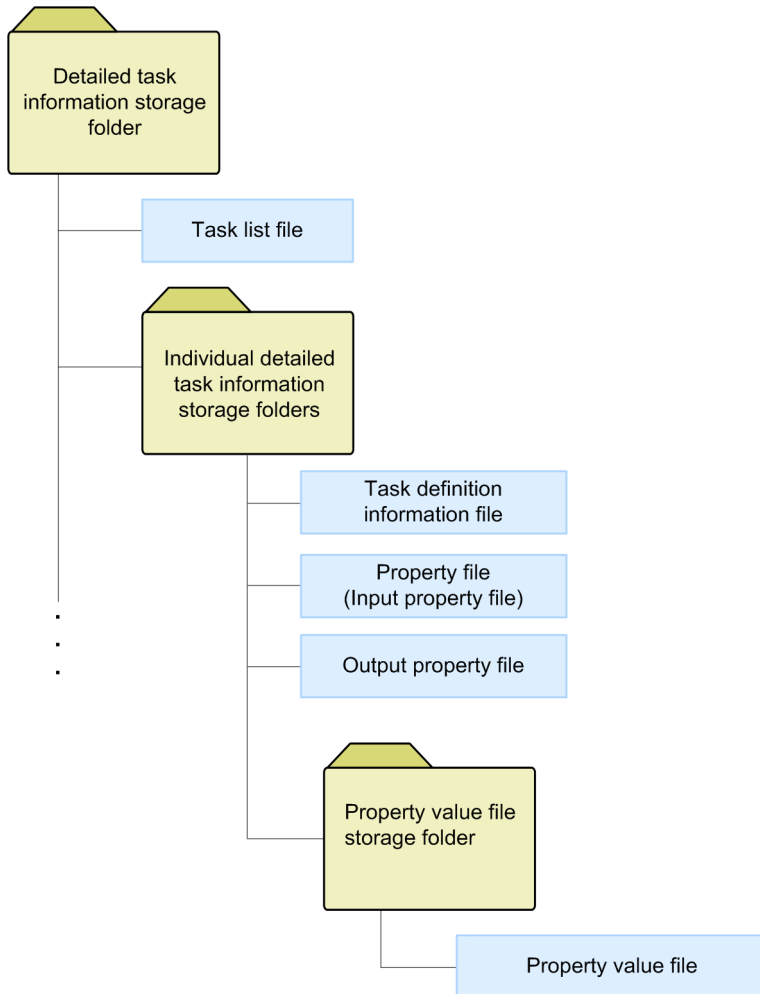
The following list shows some examples of file outputs.

```
"Task Name", "To Do", "Status", "Scheduled Time", "Start Time", "Completion
Time", "Schedule Type",
"Task ID", "Description", "Service", "Service Group", "Tags", "Submitted By", "Submit
Time", "Schedule Interval",
"Recurrence Time", "Schedule Start Date", "Notes", "Step Start Time", "Supported Action
Type", "Service Status"
"Remote command execution_20150828130932", "FALSE", "Failed", "", "2015/8/28
13:09", "2015/8/28 13:09", "immediate", "4015",
"", "Remote command
execution", "DefaultServiceGroup", "Basic, OS_Operations", "System", "2015/8/28
13:09", "", "", "", "", "", "forciblyStop, retry", "Release"
```

## When outputting the detailed task information (when `taskdetails` is specified for the output option)

The following shows the data that is output to the detailed task information storage folder.

Figure 1-2: Structure of the detailed task information storage folder

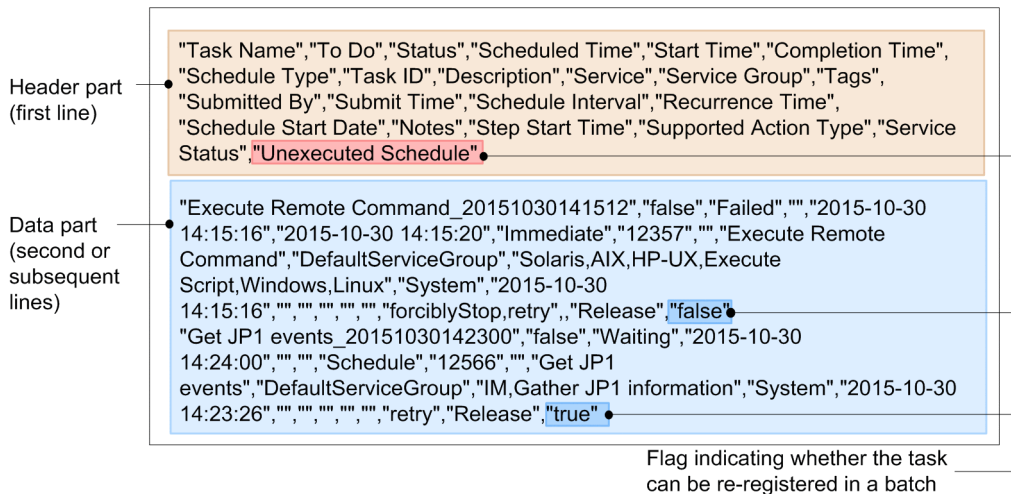


The following lists the contents of the detailed task information storage folder:

- Detailed task information storage folder  
Folder name: Arbitrary name  
The folder you specify in the `taskdetaildir` option.
- Task list file  
File name: `listtasks.csv`  
The file you can use to check the list of tasks contained in the detailed task information, and the tasks in that list to be re-registered by the `submittask` command.  
A flag is added at the end of each line of the task list that is output by specifying `tasks` for the output option to indicate whether the task is to be re-registered by the `submittask` in a batch.  
The following shows an output example of the task list.



Figure 1-3: Output example of the task list



The following describes the contents of the header part and data part:

#### Header part (first line)

Unexecuted Schedule

#### Data part (second or subsequent lines)

`true`: The task is to be re-registered in a batch.

`false`: The task is not to be re-registered in a batch.

A flag in the data part (second line or subsequent lines) becomes `true` if the task is a planned task (scheduled task or recurring task) and has not been executed yet, including the following:

- A scheduled task of which execution has not been started when the `listtasks` command is executed
- A recurring task that has not been canceled when the `listtasks` command is executed
- A scheduled task and recurring task that are being held when the `listtasks` command is executed

Note that the task list file is used for checking the tasks to be re-registered by the `submittask` in a batch. Do not edit this file.

- Individual detailed task information storage folders

Folder name: *task-ID*

This folder stores the definition information, property file (input property file), and output property file for individual tasks. The task ID becomes the folder name, and the number of folders that matches the number of output tasks are generated.

- Task definition information file

File name: `taskdef.xml`

The file in which task definition information is output in XML. Do not edit this file.

- Property file (input property file)

File name: `input.properties`

The input property information set for the task is output in `key=value` format. For details on the format of the property file (input property file), see the *JPI/Automatic Operation Administration Guide*. Note that only input properties with their visibility set to `Edit` and `Submit Window` are output. If no such input property exists, an empty file is created.

When you perform batch re-registration of the tasks, do not edit the input property file before executing the `submittask` command. If the file is edited, the command execution result is not supported by this product.

However, if you want to register the tasks with settings different from the original settings, create a copy of this file

and use the copy. After copying the file, specify items such as the scheduled date and time and task name that can be specified when executing the service, and then execute the service separately. If you want to change the input property value, edit the copied file as required, and execute the `submittask` command with the `propertyfile` option.

Note that information items (such as the host name, IP address, and user name) included in the input property values are output without processing. However, if the data type of the property is `password`, the obfuscated value is output.

- Output property file

File name: `output.properties`

The output property information set for the task is output in `key=value` format. If no applicable output property exists, an empty file is created.

Note that information items (such as the host name, IP address, and user name) included in the output property values are output without processing. However, if the data type of the property is `password`, the obfuscated value is output.

- Property value file storage folder

Folder name: `value_files`

This folder stores property value files.

- Property value file

- Name of a property value file for input properties  
`input_number#.txt`
- Name of a property value file for output properties  
`output_number#.txt`

#  
*number* is a sequential number starting with 0001, and it is obtained for each property type (input or output property).

If the input or output properties for the service include a composite type property, the value of the property is output as a text file. For details on the property value file format, see the *JP1/Automatic Operation Administration Guide*.

## Located in

In Windows:

```
JP1/AO-installation-folder\bin
```

In Linux:

```
/opt/jp1ao/bin
```

## Execute permission

Execute the command as a user with Administrator or root permissions for the OS. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

Permission required for the user specified for the `user` option depends on the argument specified for the `output` option.

When `tasks` or `histories` is specified for the `output` option (when outputting the list of tasks or histories)

The Admin, Develop, Modify, or Submit role must be set for the target service group from the user group that the user specified for the `user` option belongs to. The command does not output a list of tasks for any service groups for which none of these roles are set.

When `taskdetails` is specified for the `output` option (when outputting the detailed task information)

Specify the user who has the Admin role for the `user` option. The Admin role which allows access to the entire resource is required because the information on the entire tasks registered in JP1/AO are output, and the information is output without processing even if the input property or output property values include information such as the host name, IP address, user name, and password. Store the output detailed task information in a properly access-controlled location.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
4	One of the other commands is running.
5	Communication failed.
6	Authentication failed.
7	An invalid path is specified.
8	The file with the same name already exists in the output path.
9	The specified path does not exist.
10	The specified path is not accessible.
11	The specified folder is not empty.
12	An invalid encoding is specified.
13	Outputting the file failed.
14	You do not have permission to execute the command.
150	Obtaining the list of tasks failed.
151	Obtaining the list of histories failed.
152	Obtaining the detailed task information failed.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following examples show how to use the command for each case.

- To output, in Windows, the list of registered tasks to a file in default encoding of the OS:  
`listtasks /output tasks /file list01.csv /user user01 /password pass01`
- To output, in Windows, the tasks in the list of tasks that started or are scheduled to start from January 1, 2012 to March 31, 2012 to a file in UTF-8 encoding:  
`listtasks /startrange 2012-01-01,2012-03-31 /output histories /file list02.csv /encoding UTF-8 /user user02 /password pass02`
- To output, in Windows, the detailed task information:

```
listtasks /output taskdetails /taskdetaildir "C:\data\taskdetail" /user
user03 /password pass03
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
  - [1.6.12 submittask \(executing a service and re-registering the tasks in a batch\)](#)
  - [1.7.1 backupsystem \(backing up the JP1/AO system\)](#)
  - [1.7.6 restoresystem \(restoring the JP1/AO system\)](#)
  - [Topic Overview of property files in the JP1/Automatic Operation Administration Guide](#)
- 

## 1.6.10 setremoteconnection (adding or updating a connection destination definition)

### Description

This command registers or updates a connection destination definition by using a connection destination definition information file (in CSV format).

Before you execute this command, execute the `listremoteconnections` command, and then edit the output connection destination definition information file.

### Syntax

```
setremoteconnection
  /file connection-destination-definition-information-file-path
  /user user-ID
  {/password password | /passwordfile password-file-path}
```

### Arguments

`/file`

This option specifies the absolute or relative path to the connection destination definition information file. An error occurs if the specified file does not exist.

`/user`

This option specifies the user ID for JP1/AO.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are half-width alphanumeric characters, `!`, `#`, `$`, `%`, `&`, `'`, `(`, `)`, `*`, `+`, `-`, `.`, `=`, `@`, `\`, `^`, `_`, and `|`.

This option is not case sensitive.

`/password`

This option specifies the password of the user indicated by the `user` option.

You must specify either this option or the `passwordfile` option. If both options are specified, or if neither is specified, then you will get an error.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are the same as those for the `user` option.

/passwordfile

This option specifies the absolute or relative path to the password file for the user specified in the `user` option. You can create a password file by using the `encryptpassword` command.

You must specify either this option or the `password` option. If both options are specified, or if neither is specified, then you will get an error.

## Format of the connection destination definition information file

The following table describes the format of the connection destination definition information file.

Table 1-9: Format of the connection destination definition information file

Item	Information to be specified <sup>#1</sup>
Id	To add a connection destination definition: Specify an empty string.  To update a connection destination definition: Specify the ID of the connection destination definition containing information that you want to update.
Method	Specify one of the following values as the connection destination type. This item is not case sensitive. <ul style="list-style-type: none"><li>IPv4</li><li>IPv6</li><li>HostName</li></ul>
IP Address/Host Name	Specify the IP address or host name of the host to connect to.
Service Group	Specify the service group to be assigned to the connection destination definition.
Authentication	Specify whether to set authentication information. This item is not case sensitive. <ul style="list-style-type: none"><li>Enable Sets authentication information.</li><li>Disable Does not set authentication information.</li></ul>
Protocol <sup>#2</sup>	Specify the authentication protocol to be used for communication with the host to connect to. This item is not case sensitive. <ul style="list-style-type: none"><li>Windows</li><li>SSH</li><li>Telnet</li></ul>
SSH Authentication Method <sup>#2</sup>	If you specify <code>SSH</code> for <code>Protocol</code> , specify the authentication method to be used for communication with the host to connect to. You can also use the character string enclosed in parentheses shown below to specify this item. This item is not case sensitive. <ul style="list-style-type: none"><li>Password Authentication (PW) Password authentication</li><li>Public Key Authentication (PK) Public key authentication</li><li>Keyboard Interactive Authentication (KI) Keyboard interactive authentication</li></ul>
User ID <sup>#2</sup>	Specify the user ID for logging in to the host to connect to. If you specify <code>Windows</code> or <code>SSH</code> for <code>Protocol</code> , make sure that you also specify this item.
Password <sup>#2</sup>	Specify the password for logging in to the host to connect to. Make sure that you specify this item in the following cases: <ul style="list-style-type: none"><li>When <code>Windows</code> is specified for <code>Protocol</code></li><li>When <code>Password Authentication</code> or <code>Keyboard Interactive Authentication</code> is specified for <code>SSH Authentication Method</code></li></ul>

Item	Information to be specified#1
Password#2	In addition, the command works as follows depending on whether a value is specified for Id: When no value is specified for Id: <ul style="list-style-type: none"> <li>• If you specify ***** for Password, an error occurs.</li> </ul> When a value is specified for Id: <ul style="list-style-type: none"> <li>• If you specify ***** for Password, the password is not changed.</li> <li>• If you specify an empty character for Password, the password is deleted.</li> </ul>
Superuser's Password#2	Specify the password for the superuser of the host to connect to. Specify this item when SSH or Telnet is specified for Protocol. Note that, if you specify ***** , the password is not changed. If you specify an empty character, the password is deleted.
Connection Status	Specify the status when JP1/AO last connected to the host. <ul style="list-style-type: none"> <li>• Connection Successful  Connection was successful.</li> <li>• Error  Connection failed.</li> <li>• Unknown  Not connected</li> <li>• -  Not applicable</li> </ul>
Connected Time	Specify the time at which JP1/AO last connected to the host.

#1

Using a value without enclosing it in double quotation marks (") does not cause an error. However, if the value contains any double quotation mark ("), escape the mark with another one added in front of the mark.

#2

If you specify Disable for Authentication, specify an empty string.

## Located in

In Windows:

*JP1/AO-installation-folder\bin*

In Linux:

*/opt/jplao/bin*

## Execute permission

Execute the command as a user with both Administrator or root permissions for the OS and with the Admin role for JP1/AO. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.

Return code	Description
2	The command execution has been interrupted.
3	The service status is invalid.
4	One of the other commands is running.
5	Communication failed.
6	Authentication failed.
7	An invalid path is specified.
9	The specified path does not exist.
10	The specified path is not accessible.
14	You do not have permission to execute the command.
230	The information specified as the connection destination definition is invalid.
231	Registration of some connection destination definitions failed.
232	Registration of all connection destination definitions failed.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following example shows how to use the command to add or update, in Windows, connection destination definitions by using the information in the `list01.csv` file:

```
setremoteconnection /file list01.csv /user user01 /password pass01
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- 

## 1.6.11 stoptask (stopping a task)

### Description

This command stops execution of a task by specifying the ID of the task. However, execution of a debug task cannot be stopped.

Note that this command cannot forcibly stop execution of a task.

### Syntax

```
stoptask
  /taskid task-ID
  /user user-ID
  {/password password | /passwordfile password-file-path}
```

## Arguments

`/taskid task-ID`

This option specifies the task ID of the task of which you want to stop execution.

The possible values are half-width numeric characters (in decimal number) in 16 or fewer digits.

`/user user-ID`

This option specifies the user ID for JP1/AO.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are half-width alphanumeric characters, `!`, `#`, `$`, `%`, `&`, `'`, `(`, `)`, `*`, `+`, `-`, `.`, `=`, `@`, `\`, `^`, `_`, and `|`.

This option is not case sensitive.

`/password password`

This option specifies the password of the user indicated by the `user` option.

You must specify either this option or the `passwordfile` option. If both options are specified, or if neither are specified, then you will get an error.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are the same as those for the `user` option.

`/passwordfile password-file-path`

This option specifies the absolute or relative path to the password file for the user specified in the `user` option. You can create a password file by using the `encryptpassword` command.

You must specify either this option or the `password` option. If both options are specified, or if neither are specified, then you will get an error.

## Located in

In Windows:

`JP1/AO-installation-folder\bin`

In Linux:

`/opt/jplao/bin`

## Execute permission

Execute the command as a user with Administrator or root permissions for the OS. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

The Admin, Develop, Modify, or Submit role must be set for the service group of the target task from the user group that the user who executes the command belongs to. The command does not stop any tasks in a service group for which none of these roles are set.

## Remarks

Execute this command when the task, which you want to stop execution of, is either in In Progress, Waiting for Response, or Abnormal Detection status. If you execute this command in any other status, the command fails with the return code 140.

## Return code

The following table lists the return codes from the command.



Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
5	Communication failed.
6	Authentication failed.
7	An invalid path is specified.
9	The specified path does not exist.
14	You do not have permission to execute the command.
140	Stopping the execution of the task failed.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following example shows how to use the command to stop, in Windows, execution of the specified task (whose task ID is 1):

```
stoptask /taskid 1 /user user01 /password pass01
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
  - [Topic Stopping tasks \(execution stop\) in the JP1/Automatic Operation Administration Guide](#)
  - [Topic Stopping tasks \(forced stop\) in the JP1/Automatic Operation Administration Guide](#)
- 

## 1.6.12 submittask (executing a service and re-registering the tasks in a batch)

### Description

The functionality of the `submittask` command is as follows:

#### Executing a service

This command executes a specified service based on user-specified information such as the service name, service group name, and property values. When the task is executed normally, a message reporting the task ID is output. This command cannot execute debug services. By specifying the options, you can execute a service recursively or at a specified execution date and time. If you do not specify any options, the command executes the service immediately.

#### Re-registering the tasks in a batch

This command re-registers the scheduled tasks and recurring tasks in a batch based on the contents of the detailed task information output by the `listtasks` command. The re-registered task inherits the settings and conditions from the original task. This command is a functionality for executing a scheduled or recurring service with the same settings by

referring to the information stored in the detailed task information storage folder. Note that this command is not a functionality for restoring the same task. The re-registered task is, therefore, a task different from the original task and has a different task ID. Debug tasks cannot be re-registered.

#### Procedures before re-registering the tasks in a batch

Perform the following before re-registering the tasks in a batch:

- The detailed task information storage folder must be the folder output by JP1/AO whose version and revision is the same as those of JP1/AO you use to re-register the tasks in a batch. Batch re-registration of tasks fails if the detailed task information storage folder output by JP1/AO whose version or revision is different is specified.
- Check that the detailed task information storage folder output by the `listtasks` command exists.
- Set up the definition information (service, service template, user, user group, service group, connection destinations, and service share properties) and definition file separately. Restore those settings by using the `backupsystem` and `restoresystem` commands if necessary. Do not change those settings or delete any service after outputting the detailed task information storage folder by the `listtasks` command. If the service settings have been changed, batch re-registration is performed according to the changed settings. If the service settings have been deleted, re-registration of the corresponding task fails.
- For a scheduled task, confirm that the specified time has not been passed at the time of task re-registration. An error occurs if the specified time has passed, and you cannot directly re-register the task.

#### Condition of the tasks that are re-registered in a batch

The tasks that are re-registered in a batch are the unexecuted scheduled tasks and recurring tasks that are in the task list (`listtasks.csv`) in the detailed task information storage folder. In the task list, the `Unexecuted Schedule` column of the unexecuted scheduled tasks and recurring tasks is `true`.

#### Re-registering a scheduled task of which scheduled time has passed

You cannot directly re-register a task of which scheduled time has passed. If you re-register the task in a batch, re-registration fails with a message indicating that the specified date and time has passed. To check the settings of the task of which re-registration failed, refer to the task list file (`listtasks.csv`) in the detailed task information storage folder, and the property file (input property file) and output property file in the individual detailed task information storage folder. If you want to register a task of which scheduled time has passed, check the original date and time in the task list (`listtasks.csv`) in the detailed task information storage folder, and then execute each service by specifying a new date and time using the **Service** window or the `submittask` command of JP1/AO. Note that the start time must be equal to or after the current date and time.

#### Measures to take when there is a task of which re-registration failed

If batch re-registration of tasks fails, a message indicating that task registration failed, and task IDs of the tasks of which registration failed are displayed. These task IDs are the ones output by the `listtasks` command. If some tasks are successfully re-registered, move the individual detailed task information storage folder for the relevant tasks to another location. Then eliminate the causes of the failure, and execute the command again. Moving the folders is to prevent duplicate registration of the successful tasks. If the same error occurs after taking the above measures, contact the system administrator.

## Syntax

The syntax of the `submittask` command is as follows:

### When executing a service immediately

```
submittask
  /servicename service-name
  [/servicegroup service-group-name]
```

```
[/taskname task-name]
[/taskdescription task-description]
[/property property-key property-value |
 /propertyfile property-file-path]
/user user-ID
{/password password | /passwordfile password-file-path}
[/wait]
```

## When executing a service at a specified date and time

```
submittask
 /servicename service-name
[/servicegroup service-group-name]
[/taskname task-name]
[/taskdescription task-description]
[/property property-key property-value |
 /propertyfile property-file-path]
/user user-ID
{/password password | /passwordfile password-file-path}
/scheduledate yyyy-mm-dd /schedulesettime hh:mm
```

## When executing a service recursively

```
submittask
 /servicename service-name
[/servicegroup service-group-name]
[/taskname task-name]
[/taskdescription task-description]
[/property property-key property-value |
 /propertyfile property-file-path]
/user user-ID
{/password password | /passwordfile password-file-path}
/recurrencepattern {daily[:{1h|2h|3h|4h|6h|8h|12h|24h}] |
 weekly:sun,mon,...,sat | monthly:{dd,dd,...,dd[,endofmonth]} | endofmonth}}
/recurrencestart hh:mm /recurrencestart yyyy-mm-dd
```

## When re-registering the tasks in a batch

```
submittask
 /reregister
 /taskdetaildir detailed-task-information-storage-folder
[/setoriginalsubmitter]
/user user-ID
{/password password | /passwordfile password-file-path}
```

## Arguments

*/servicename service-name*

This option specifies the name of the service to be performed.

The number of possible characters is in the range from 1 to 128 characters.

*/servicegroup service-group-name*

This option specifies the name of the service group that the service to be performed belongs to.

If this option is omitted, the service group associated with the user specified in the argument is used. However, if more than one service group is associated with that user, an error occurs.

The number of possible characters is in the range from 1 to 80 characters. The possible characters are any characters other than the unicode characters from U+10000 to U+10FFFF.

Note that, instead of the `servicegroup` option, you can also specify the service group name by using the `resourcegroup` option, which was used in JP1/AO 10-52 and earlier. If you specify `All Resources` for the `servicegroup` option, the service will run as if `DefaultServiceGroup` is specified.

`/taskname task-name`

This option specifies the name of the task.

If this option is omitted, the system uses `service-name_YYYYMMDDhhmmss` (where `YYYYMMDDhhmmss` is the time when the service is performed) as a default name.

The number of possible characters is in the range from 1 to 128 characters. The possible characters are any characters other than the control characters (from `\u0000` to `\u001F` and from `\u007F` to `\u009F`).

`/taskdescription task-description`

This option specifies the description of the task.

If this option is omitted, the value is not set.

The number of possible characters is in the range from 1 to 256 characters. The possible characters are any characters other than the control characters (from `\u0000` to `\u001F` and from `\u007F` to `\u009F`).

`/property property-key property-value`

This option specifies the property key and value that the service to be performed uses. The system verifies whether the specified property value is valid according to the service template specifications.

For property keys that are not set in this option, the values specified in the **Service Definition** window will be used. If values for required properties are not specified in either the **Service Definition** window or by this option, an error occurs.

You can use multiple instances of this option to specify multiple property key and value combinations (format: `/property key-1 value-1 /property key-2 value-2 ...`). By default, you can specify a maximum of 1,000 instances of this option. You can specify the maximum number of properties that can be specified by using the user-specified properties file (`config_user.properties`).

- *property-key*

This option specifies the property key for the service.

The number of possible characters is in the range from 1 to 1,024 characters. The possible characters are half-width alphanumeric characters, `-`, `_`, `.`, and `/`.

If the same property key is specified more than once, then an error occurs.

- *property-value*

This option specifies the property value for the property key.

Any value containing a space or special character must be enclosed in double quotation marks (`"`).

`/propertyfile property-file-path`

This option specifies the absolute or relative path to the property file, which defines the input property settings that the service to be performed uses.

For property keys and property values that are not set in the property file specified by this option, the values specified in the **Service Definition** window (Create, Edit, or Copy) or the **Submit Service** window will be used. If values for required properties are not specified in either the **Service Definition** window (Create, Edit, or Copy) or in the **Submit Service** window, and the values are not defined in the property file specified by this option, an error occurs.

For details on the format of the property file, see the *JP1/Automatic Operation Administration Guide*.

The following table shows the format of the property file.

#### `/reregister`

Specify this option if you re-register the tasks in a batch. Make sure that you also specify the `taskdetaildir` option when you specify the `reregister` option.

#### `/taskdetaildir detailed-task-information-storage-folder`

This option is required if the `reregister` option is specified. This option specifies the absolute or relative path to the detailed task information storage folder that stores the scheduled or recurring task information you want to re-register. Note that only a folder on the local disk can be specified. The number of characters that can be specified for the absolute path is no more than 190 characters.

#### `/setoriginalsubmitter`

If you specify this option when re-registering the tasks in a batch, the task submitter after re-registration displays the name of the user who submitted the original task, not the user who re-registered the task. The user who submitted the original task is the user who was executing the task at the time when the `listtasks` command was used to output the detailed task information. You can check the task submitter after re-registration from the user ID displayed in the **Submitted By** column in the **Tasks** window. You can check the user who was executing the task at the time when the `listtasks` command was used to output the detailed task information in the `Submitted By` column in the `listtasks.csv` file that is output in the detailed task information storage folder.

If you omit this option, the user ID specified for the `user` option of the `submittask` command becomes the task submitter after re-registration.

Note that an error does not occur even if "the user who was executing the task at the time when the `listtasks` command is used to output the detailed task information" does not exist when re-registering the task. In this case, the task submitter becomes "the user who is executing the task at the time when the `listtasks` command is used to output the detailed task information".

#### `/user user-ID`

This option specifies the user ID for JP1/AO. Make sure that you specify the ID of a user that is associated with a service group that the service specified by the `servicename` option belongs to.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are half-width alphanumeric characters, `!`, `#`, `$`, `%`, `&`, `'`, `(`, `)`, `*`, `+`, `-`, `.`, `=`, `@`, `\`, `^`, `_`, and `|`.

This option is not case sensitive.

#### `/password password`

This option specifies the password of the user indicated by the `/user` option.

The number of possible characters is in the range from 1 to 256 characters.

The possible characters are the same as those for the `user` option.

#### `/passwordfile password-file-path`

This option specifies the absolute or relative path to the password file for the user specified in the `user` option. You can create a password file by using the `encryptpassword` command.

#### `/wait`

If this option is specified, the command outputs the task execution result (Completed or Failed), and then terminates.

If the `wait` option is not specified, the command terminates without waiting for the task to terminate. In this case, a message reporting the task ID is output only when the task execution has started normally.

#### `/scheduledate`

If you want to execute the service according to a schedule, specify the date (year, month, and day) that the service will be executed in the `YYYY-MM-DD` format. In `YYYY`, specify a four-digit year. In `MM`, specify a month number from 1 (or 01) to 12. In `DD`, specify a day number from 1 (or 01) to 31. Note that when you specify the `scheduledate` option, you must also specify the `scheduletime` option. The command execution will fail if:

- The combination of arguments is invalid.  
For details on the combination of arguments, see [Table 1-10: Argument combination of the `submittask` command](#).
- The date is specified in an incorrect format.
- The execution time determined by the combination of this option and the `scheduletime` option is earlier than the current time.
- The specified date is not within the range from 1994-01-01 to 2036-12-31.

`/scheduletime`

If you want to execute the service according to a schedule, specify the time (hour and minute) in the `hh:mm` format. In `hh`, specify the hour from 00 to 23. In `mm`, specify the minute from 00 to 59. When you specify the `scheduletime` option, you must also specify the `scheduledate` option. The command execution will fail if:

- The combination of arguments is invalid.  
For details on the combination of arguments, see [Table 1-10: Argument combination of the `submittask` command](#).
- The time is specified in an incorrect format.
- The execution time determined by the combination of this option and the `scheduledate` option is earlier than the current time.

`/recurrencepattern {daily[:{1h|2h|3h|4h|6h|8h|12h|24h}] | weekly:sun,mon,...,sat | monthly:{dd,dd,...,dd[,endofmonth] | endofmonth}`

This option specifies the recurrence pattern of the service execution. When you specify the `recurrencepattern` option, you must also specify the `/recurrencetime` and `recurrencestart` options. Note that the command execution fails if either of the following conditions applies:

- The combination of arguments is invalid.  
For details on the combination of arguments, see [Table 1-10: Argument combination of the `submittask` command](#).
- The specified recurrence pattern is in an invalid format.

There are three types of recurrence pattern: daily, weekly, and monthly. The format of the recurrence pattern differs by the recurrence pattern type.

#### Daily

Specify `daily` to execute the command once a day.

To specify the recurrence interval in hours, specify in the following format: `daily:{1h|2h|3h|4h|6h|8h|12h|24h}`. Start with `daily:`, and then select the time interval from 1h, 2h, 3h, 4h, 6h, 8h, 12h, and 24h.

#### Weekly

Specify the pattern in the `weekly:sun,mon,...,sat` format.

Preceded by `weekly:`, specify one or more days on which you want to execute the service, delimiting them by a comma (.). To specify days in the abbreviated form, use `sun, mon, tue, wed, thu, fri, and sat`. The order of the specified days does not matter. An invalid argument error occurs if the same day is specified for multiple times.

#### Monthly

Specify the pattern in the `monthly:{dd,dd,...,dd[,endofmonth] | endofmonth}` format.

Specify `monthly:` followed by one or more dates on which to execute the services, with the dates delimited by commas. To execute the service at the end of the month, specify `endofmonth`. You can specify the dates in any order. If you want to specify execution at the end of the month in addition to specific dates, specify

endofmonth at the end of the sequence. Specify dates as single-byte numerals in the range from 1 (or 01) to 31. In the following circumstances, an invalid argument error occurs:

- The same date is specified multiple times
- A nonexistent date such as 0 or below or 32 or above is specified
- endofmonth is specified but not at the end of the sequence

Note that the service will not be executed in a month that does not contain the specified date. For example, if the task is scheduled to be executed on the 30th or 31st of every month, recurring execution of that task will be skipped in February.

`/recurrence time hh:mm`

This option specifies the time (hour and minute) at which to execute the service in *hh:mm*. For *hh*, specify the hour from 00 to 23. For *mm*, specify the minute from 00 to 59. When you specify the `recurrence time` option, you must also specify the `recurrence pattern` and `recurrence start` options. Note that the command execution fails if either of the following conditions applies:

- The combination of arguments is invalid.  
For details on the combination of arguments, see [Table 1-10: Argument combination of the `submit task` command](#).
- The specified time is in an invalid format.

`/recurrence start yyyy-mm-dd`

This option specifies the date on which to start executing the recurring service in *yyyy-mm-dd*. For *yyyy*, specify the year in four digits. For *mm*, specify the month from 1 (or 01) to 12. For *dd*, specify the date from 1 (or 01) to 31. When you specify the `recurrence start` option, you must also specify the `recurrence pattern` and `recurrence time` options. Note that the command execution fails if one of the following conditions applies:

- The combination of arguments is invalid.  
For details on the combination of arguments, see [Table 1-10: Argument combination of the `submit task` command](#).
- The specified date is in an invalid format.
- The specified date is out of the following range: from 1/1/1994 to 12/31/2036.

## Argument combination of the `submit task` command

Table 1-10: Argument combination of the `submit task` command

Option	Immediate execution of the service	Scheduled execution of the service	Recurring execution of the service	Re-registration of the scheduled tasks in a batch
<code>/servicename</code>	Required	Required	Required	--
<code>/servicegroup</code>	Optional	Optional	Optional	--
<code>/taskname</code>	Optional	Optional	Optional	--
<code>/taskdescription</code>	Optional	Optional	Optional	--
<code>/property#1</code>	Optional	Optional	Optional	--
<code>/propertyfile#1</code>	Optional	Optional	Optional	--
<code>/reregister</code>	--	--	--	Required
<code>/taskdetaildir</code>	--	--	--	Required
<code>/setoriginalsubmitter</code>	--	--	--	Optional



Option	Immediate execution of the service	Scheduled execution of the service	Recurring execution of the service	Re-registration of the scheduled tasks in a batch
/user	Required	Required	Required	Required
/password#2	Required	Required	Required	Required
/passwordfile#2	Required	Required	Required	Required
/wait	Optional	--	--	--
/scheduledate	--	Required	--	--
/schedulesettime	--	Required	--	--
/recurrencepattern	--	--	Required	--
/recurrencetime	--	--	Required	--
/recurrencestart	--	--	Required	--

#### Legend:

Required: Required. An argument error occurs if omitted.

Optional: Can be omitted.

--: Cannot be specified. An argument error occurs if specified.

#1

Specify either the `property` option or `propertyfile` option. An error occurs if you specify both options at the same time.

#2

Specify either the `password` option or `passwordfile` option. An error occurs if you specify both options at the same time.

## Located in

In Windows:

`JP1/AO-installation-folder\bin`

In Linux:

`/opt/jplao/bin`

## Execute permission

Execute the command as a user with Administrator or root permissions for the OS. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

Execute the command as a user with Administrator permissions for the OS. If a user without Administrator permissions executes the command, a message appears asking the user to elevate the permission level.

Before the service can be executed, make sure that the Admin, Develop, Modify, or Submit role is set for the resource group of that service from the user group that the user who executes the command belongs to. The command cannot execute a service in a resource group for which none of these roles are set.

The following describes the permission required for the user specified for the `user` option.



## When executing a service

The Admin, Develop, Modify, or Submit role must be set for the target resource group from the user group that the user specified for the `user` option belongs to. The user can only execute a service for which he or she has the execute permission.

## When re-registering a task in a batch

The Admin role must be set for the user specified for the `user` option.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
5	Communication failed.
6	Authentication failed.
7	An invalid path is specified.
9	The specified path does not exist.
10	The specified path is not accessible.
14	You do not have permission to execute the command.
130	Starting the service failed.
131	The property file does not exist.
132	The property file has an invalid format.
133	The status of the task could not be obtained (when the <code>wait</code> option is specified).
134	The task could not be executed (when the <code>wait</code> option is specified).
136	The data format of the detailed task information storage folder is invalid.
137	Re-registering the planned tasks in a batch partially failed.
138	Re-registering the planned tasks in a batch failed entirely.
139	The version or revision of JP1/AO that was used to output the detailed task information storage folder is different from the currently installed JP1/AO.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following examples show how to use the command for each case.

- To execute, in Windows, a service specified by the service name with the property keys and values:  
`submittask /servicename service01 /user user01 /password pass01 /property keyA valueA /property keyB "value B" /property keyC valueC,valueD`

- To execute, in Windows, a service specified by the service group and the service name, with the task name, task description, and property file:  

```
submittask /servicename service02 /servicegroup servicegroupA /taskname task02 /taskdescription testtask /propertyfile C:\properties.txt /user user02 /password pass02
```
- To output, in Windows, the task execution result before the command terminates:  

```
submittask /servicename service03 /user user03 /password pass03 /wait
```
- To execute, in Windows, a service at a specified time:  

```
submittask /servicename service04 /user user04 /password pass04 /scheduledate 2014-01-01 /scheduleset 15:30
```
- To execute, in Windows, a service recursively:  

```
submittask /servicename service05 /user user05 /password pass05 /recurrencepattern weekly:sun,mon,fri /recurrencetime 15:30 /recurrencestart 2013-06-17
```
- To re-register, in Windows, planned tasks in a batch:  

```
submittask /reregister /taskdetaildir "C:\data\taskdetail" /user user06 /password pass06
```

---

## Related topics

- [1.6.9 listtasks \(outputting the list of tasks and the detailed task information\)](#)
  - [1.3 Valid characters for arguments in a command](#)
  - [Topic User-specified properties file \(config\\_user.properties\) in the JP1/Automatic Operation Configuration Guide](#)
  - [Topic Overview of property files in the JP1/Automatic Operation Administration Guide](#)
-

## 1.7 Maintenance-related commands

---

### 1.7.1 backupsystem (backing up the JP1/AO system)

#### Description

This command backs up the configuration and database information of JP1/AO to store the data in the specified folder.

#### Syntax

```
backupsystem
  /dir backup-data-path
  [/auto]
```

#### Arguments

*/dir backup-data-path*

This option specifies the absolute or relative path to an empty folder in which backup data is collected and stored. A folder in the local disk drive can only be specified. We recommend that you use a path that has 50 or fewer characters.

*/auto*

This option causes the command to automatically start and stop the services and databases of JP1/AO, JP1/OA, and the Hitachi Command Suite products. If this option is omitted, the services and databases of JP1/AO, JP1/OA, and the Hitachi Command Suite products are not automatically started and stopped.

If you want to use this option in a cluster environment, services registered with the cluster software must be offline.

#### Located in

In Windows:

*JP1/AO-installation-folder\bin*

In Linux:

*/opt/jp1ao/bin*

#### Execute permission

Execute the command as a user with Administrator or root permissions. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

#### Remarks

- Make sure that the folder in which backup files are stored has a sufficient free space. The required free space is as follows:  
Total size of the files to be backed up + 20 MB
- This command does not back up the following files (manual backup, as necessary, is required).
  - SSL server certificate file for HTTPS connection
  - Private key file for HTTPS connection
  - Private key file for public key authentication

- Cluster service control command that was created (in a Linux cluster configuration)
- If you do not specify the `auto` option, then make sure that the JP1/AO services are not running before executing this command.  
If the services are still running, execute the `hcnds64srv` command with the `stop` option to stop the services.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
4	One of the other commands is running.
7	An invalid path is specified.
9	The specified path does not exist.
10	The specified path is not accessible.
11	The specified folder is not empty.
14	You do not have permission to execute the command.
100	Performing the backup failed.
101	Starting or stopping the service failed.
103	An access to the scheduler database failed.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following example shows how to use the command to back up, in Windows, data in the specified backup folder (C:\Users\Backup):

```
backupsystem /dir "C:\Users\Backup" /auto
```

## Related topics

- [1.3 Valid characters for arguments in a command](#)

## 1.7.2 hcnds64dbrepair (re-creating the database)

### Description

This command forces all the databases to be deleted, re-creates them, and then recovers them using the backup data obtained by the `hcnds64dbtrans` command. You use this command if any of the databases is corrupted and using the `restoresystem` command and the `hcnds64dbtrans` command with the `import` option specified cannot restore the database.

## Syntax

```
hcms64dbrepair  
  /trans backup-data
```

## Arguments

`/trans backup-data`

This option specifies the backup data obtained using the `hcms64dbtrans` command. Make sure that you specify the path specified in the `/workpath` or `file` option of the `hcms64dbtrans` command.

## Located in

In Windows:

`Common-Component-installation-folder\bin`

In Linux:

`/opt/HiCommand/Base64/bin`

## Execute permission

Execute the command as a user with Administrator or root permissions.

## Remarks

- Stop the JP1/AO system before executing the `hcms64dbrepair` command.
- Start the JP1/AO system after executing the `hcms64dbrepair` command.
- The command uses the `Common-Component-installation-folder\tmp` folder or the `var/opt/HiCommand/Base64/tmp` folder to extract the backup data. Secure enough space to extract the backup data according to the size of the data.
- After the command execution, the password of the built-in account (System account) is initialized. Change the password if necessary.
- In a cluster system, execute this command on the executing host. This command cannot be executed on the standby host.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
245	Importing the database failed.
246	The definition file is invalid.
247	An attempt to undo setup of the database failed.
248	Stopping a service or database failed.
249	The command cannot be executed on the standby node.

Return code	Description
250	The backup data is invalid. (Some files are missing or extracting the archive file failed.)
251	The command has been interrupted due to inconsistency in the product or product version.
252	Setting up the database failed.
253	Starting the service for database failed.
254	The database cannot be re-created due to its incomplete setup.
255	The command terminated abnormally.

## Example

The following example shows how to use the command to force all the databases to be deleted, re-create them, and then recover them by using backed up data, in Windows:

```
hcnds64dbrepair /trans C:\bkfile1
```

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- [Topic Starting a JP1/AO system \(non-cluster configuration\) in the JP1/Automatic Operation Administration Guide](#)
- [Topic Starting a JP1/AO system \(cluster configuration\) in the JP1/Automatic Operation Administration Guide](#)
- [Topic Stopping a JP1/AO system \(non-cluster configuration\) in the JP1/Automatic Operation Administration Guide](#)
- [Topic Stopping a JP1/AO system \(cluster configuration\) in the JP1/Automatic Operation Administration Guide](#)
- [1.6.4 hcnds64srv \(starting and stopping JP1/AO, and displaying the status of JP1/AO\)](#)

## 1.7.3 hcnds64dsrv (starting and stopping the databases)

### Description

This command starts and stops the databases of JP1/AO. You use this command when maintaining the databases.

### Syntax

```
hcnds64dsrv
  {/start | /stop}
```

### Arguments

`/start`

This option causes the command to start the databases.

`/stop`

This option causes the command to stop the databases.

## Located in

In Windows:

```
Common-Component-installation-folder\bin
```

In Linux:

```
/opt/HiCommand/Base64/bin
```

## Execute permission

Execute the command as a user with Administrator or root permissions.

## Remarks

This command is restricted for database maintenance procedures.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The system accepted the start or stop request.
254	The databases are not initialized.
255	The command execution failed.

## Example

The following examples show how to use the command for each case.

- To start, in Windows, the databases of JP1/AO:  
`hcnds64dbsrv /start`
- To stop, in Windows, the databases of JP1/AO:  
`hcnds64dbsrv /stop`

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- 

## 1.7.4 hcnds64dbtrans (backing up and restoring the databases)

### Description

This command backs up (exports) or restores (imports) the databases of JP1/AO. You use this command when re-organizing the databases of JP1/AO.

## Syntax

To back up (export) the databases of JP1/AO:

```
hcnds64dbtrans
/export
/workpath working-folder-path
/file archive-file-path
[/auto]
```

To restore (import) the databases of JP1/AO:

```
hcnds64dbtrans
/import
/type Automation
/workpath working-folder-path
[/file archive-file-path]
[/auto]
```

## Arguments

`/export`

This option causes the command to export the databases.

`/workpath working-folder-path`

This option specifies the absolute path to a working folder that is temporarily used for exporting or importing. A folder on the local disk drive can only be specified.

Use an empty folder for the working folder when you specify the `/file` option for exporting or importing.

`/file archive-file-path`

This option specifies the absolute path to the archive file to which the data is exported or from which the data is imported. This option is required if the `export` option is specified.

The archive file is not created if the output file size exceeds 2 GB, or if the amount of disk space for a location in which the archive file is created is insufficient.

`/auto`

This option causes the command to automatically start and stop the services and databases of JP1/AO, JP1/OA, and the Hitachi Command Suite products. If this option is omitted, the services and databases of JP1/AO, JP1/OA, and the Hitachi Command Suite products are not automatically started and stopped.

`/import`

This option causes the command to import the databases. All the existing authentication data is deleted before the data is imported.

`/type Automation`

This option specifies `Automation` as the name of the product whose database is to be imported.

## Located in

In Windows:

`Common-Component-installation-folder\bin`

In Linux:

`/opt/HiCommand/Base64/bin`



## Execute permission

Execute the command as a user with Administrator or root permissions.

## Remarks

- If the return code 3 is output by an export operation, the database information remains in the directory specified for the `workpath` option.

To import this information, set the directory that you specified for the `workpath` option at the time of the export operation for the `workpath` option for the import operation. At this time, do not change the folder structure in the directory you specified for the `workpath` option at the time of the export operation. In addition, do not specify any value for the `file` option when performing the import operation.

- In the following cases, the directory specified for the `/workpath` option becomes empty, and the command is completed.
  - When the return code 1, 2, 233, 234, 235, 237, 238, 239, 240, or 255 is output by an export operation
  - When the return code 3 is output by an import operation

## Return code

The following table lists the return codes from the command with the `export` option.

Return code	Description
0	The command succeeded.
1	Obtaining the product version failed.
2	The databases are not running.
3	Archiving the databases failed.
4	The working folder is not empty.
233	Restarting the databases is being interrupted.
234	The database services are stopped or do not exist.
235	The databases are not initialized.
237	Starting the Hitachi Command Suite products or databases failed.
238	Stopping the Hitachi Command Suite products or databases failed.
239	Starting the databases failed.
240	Stopping the databases failed.
255	The command terminated abnormally.

The following table lists the return codes from the command with the `import` option.

Return code	Description
0	The command succeeded.
1	Obtaining the product version failed.
2	The databases are not running.
3	Extracting the archive file failed.

Return code	Description
4	The working folder is not empty.
5	The specified product is not included in the archive file.
6	The specified product is not installed.
7	A version of the product that cannot be imported is found.
8	The working folder has no data to be imported, or the data for importing has an invalid format.
9	You attempted to import the data on the secondary server into the primary server.
10	You attempted to import the data on the primary server into the secondary server.
11	You attempted to import the data into the database in use.
233	Restarting the databases is being interrupted.
234	The database services are stopped or do not exist.
235	The databases are not initialized.
237	Starting the Hitachi Command Suite products or databases failed.
238	Stopping the Hitachi Command Suite products or databases failed.
239	Starting the databases failed.
240	Stopping the databases failed.
255	The command terminated abnormally.

## Example

The following examples show how to use the command for each case.

- To back up, in Windows, the databases of JP1/AO:

```
hcnds64dbtrans /export /workpath "C:\Users\workfolder" /file "C:\backup\arcfile01" /auto
```

- To restore, in Windows, the databases of JP1/AO:

```
hcnds64dbtrans /import /type Automation /workpath "C:\Users\workfolder" /file "C:\backup\arcfile01" /auto
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- 

## 1.7.5 hcnds64getlogs (collecting log information)

### Description

This command collects log information recorded during JP1/AO operation to output the information to the archive file.

### Syntax

```
hcnds64getlogs
  /dir output-folder-path
```

```
[/types Automation]
[/arc archive-file-name]
[/logtypes {log | db | csv}]
```

## Arguments

`/dir output-folder-path`

This option specifies the path to the folder in which the archive file is output. A folder on the local disk drive can only be specified.

You must specify the absolute or relative path to an empty folder for *output-folder-path*. If the specified folder path does not exist, then that folder is newly created.

The maximum length of the path name is 100 characters. The system grants write permission to the folder specified by this option.

`/types Automation`

This option specifies `Automation` as the name of the product whose log information is to be collected. If the JP1/AO server OS is Windows, this option is not case sensitive. If the JP1/AO server OS is Linux, the option is case sensitive. If this option is omitted, the command has an effect on JP1/AO and all the installed Hitachi Command Suite products. Note that, in this case, it might take longer to collect log information.

`/arc archive-file-name`

This option specifies the name of the archive file created as a result of execution of the data collection tool for Common Component. If this option is not specified, the archive file is named `HiCommand_log`.

The archive file is output under the folder specified in the `/dir` option.

The possible characters for the archive file name are printable ASCII characters (ranged from 0x20 to 0x7E in the ASCII code) except for some of the special characters (`\`, `/`, `:`, `,`, `;`, `*`, `?`, `"`, `<`, `>`, `|`, `$`, `%`, `&`, `'`, and ```). The extension is not necessary.

`/logtypes {log | db | csv}`

This option specifies the type of a log file for Common Component that you want to collect. The following table lists the relationship between the log file type and the log files that can be collected.

Table 1-11: Log file types and log files that can be collected

Log file type	Log file that can be collected (Windows)	Log file that can be collected (Linux)
log	<ul style="list-style-type: none"> <li><i>archive-file-name-specified-in-the-arc-option.jar</i></li> <li><i>archive-file-name-specified-in-the-arc-option.hdb.jar</i></li> </ul>	<ul style="list-style-type: none"> <li><i>archive-file-name-specified-in-the-arc-option_64.jar</i></li> <li><i>archive-file-name-specified-in-the-arc-option_64.hdb.jar</i></li> </ul>
db	<i>archive-file-name-specified-in-the-arc-option.db.jar</i>	<i>archive-file-name-specified-in-the-arc-option_64.db.jar</i>
csv	<i>archive-file-name-specified-in-the-arc-option.csv.jar</i>	<i>archive-file-name-specified-in-the-arc-option_64.csv.jar</i>

If this option is omitted, the system collects all the log files for Common Component. Because of this, we recommend that you execute the command without this option.

You can specify multiple log file types by entering them separated by half-width space characters such as `logtypes log db csv`. If you use the `/types` and `logtypes` options at the same time, you must specify `log` for the `logtypes` option.

## Output format

The table below shows the list of data collected by the command.

Note that the file content and output format are not disclosed.

Table 1-12: List of data to be collected (when the JP1/AO server OS is Windows)

Archive file	Output results
<i>output-destination-folder-specified-in-the-dir-option\Automation_1st_log.jar</i>	<ul style="list-style-type: none"> <li>All files directly under <i>JP1/AO-installation-folder\logs</i> (Subfolders are not included.)</li> <li>All files in <i>JP1/AO-installation-folder\data\task</i></li> </ul>
<i>output-destination-folder-specified-in-the-dir-option\Automation_log.jar</i>	<ul style="list-style-type: none"> <li>FILELIST.txt</li> <li>All files in <i>JP1/AO-installation-folder\conf</i></li> <li>All files in <i>JP1/AO-installation-folder\data</i></li> <li>All files in <i>JP1/AO-installation-folder\logs</i></li> <li>All files in <i>JP1/AO-installation-folder\work</i></li> <li>All files in <i>Windows-folder<sup>#1</sup>\Temp\HITACHI_JP1_INST_LOG</i></li> <li>All files in <i>Windows-folder<sup>#1</sup>\Temp\jplcommon</i></li> <li><i>Program-Files-folder<sup>#2</sup>\InstallShield Installation Information\{C4F6D00E-A9A2-4E57-A21A-B78B63FF1C54}\setup.ini</i></li> <li><i>Program-Files-folder<sup>#2</sup>\InstallShield Installation Information\{C4F6D00E-A9A2-4E57-A21A-B78B63FF1C54}\setup.ilg</i></li> <li>REGDATA.DAT</li> </ul>
<i>output-destination-folder-specified-in-the-dir-option\archive-file-name-specified-in-the-arc-option.jar</i>	Execution result of the data collection tool for Common Component (hcmds64getlogs, hcmds64ras)
<i>output-destination-folder-specified-in-the-dir-option\archive-file-name-specified-in-the-arc-option.hdb.jar</i>	Execution result of the data collection tool for Common Component (hcmds64getlogs)
<i>output-destination-folder-specified-in-the-dir-option\archive-file-name-specified-in-the-arc-option.db.jar</i>	Execution result of the data collection tool for Common Component (hcmds64getlogs)
<i>output-destination-folder-specified-in-the-dir-option\archive-file-name-specified-in-the-arc-option.csv.jar</i>	Execution result of the data collection tool for Common Component (hcmds64getlogs)

#1:

The *Windows-folder* is defaulted to C:\WINDOWS.

#2:

The *Program-Files-folder* is defaulted to C:\Program Files.

Table 1-13: List of data to be collected (when the JP1/AO server OS is Linux)

Archive file	Output results
<i>output-destination-folder-specified-in-the-dir-option/archive-file-name-specified-in-the-arc-option_64.jar</i>	<ul style="list-style-type: none"> <li>FILELIST.txt</li> <li>All files in <i>/opt/jplao/conf</i></li> <li>All files in <i>/var/opt/jplao/data</i></li> <li>All files in <i>/var/opt/jplao/logs</i></li> <li>All files in <i>/opt/jplao/tools</i></li> </ul>

Archive file	Output results
<i>output-destination-folder-specified-in-the-dir-option/archive-file-name-specified-in-the-arc-option_64.jar</i>	<ul style="list-style-type: none"> <li>All files in /var/opt/jplao/work</li> <li>All files in /tmp/ HITACHI_JP1_INST_LOG</li> <li>Execution result of the data collection tool for Common Component (hcnds64getlogs, hcnds64ras)</li> </ul>
<i>output-destination-folder-specified-in-the-dir-option/archive-file-name-specified-in-the-arc-option_64.hdb.jar</i>	Execution result of the data collection tool for Common Component (hcnds64getlogs)
<i>output-destination-folder-specified-in-the-dir-option/archive-file-name-specified-in-the-arc-option_64.db.jar</i>	Execution result of the data collection tool for Common Component (hcnds64getlogs)
<i>output-destination-folder-specified-in-the-dir-option/archive-file-name-specified-in-the-arc-option_64.csv.jar</i>	Execution result of the data collection tool for Common Component (hcnds64getlogs)

## Located in

In Windows:

*Common-Component-installation-folder\bin*

In Linux:

*/opt/HiCommand/Base64/bin*

## Execute permission

Execute the command as a user with Administrator or root permissions.

## Remarks

- Do not interrupt this command while it is running.
- If the hcnds64getlogs command is interrupted, this command has terminated before this command completed due to insufficient free space in the folder specified in the dir option. In this case, make sure that the folder has enough free space, and then execute this command again.
- Do not execute more than one hcndsgetlogs command at the same time.
- When JP1/AO is running in a cluster configuration, execute this command on both the active host and standby host. You can execute this command even if the JP1/AO server is not running. Therefore, even if an error occurs in a cluster configuration, you can collect log information without switching nodes. However, if the database is not running, you cannot obtain the database information.
- If the same option is specified more than once, only the first option is effective.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command terminated abnormally.

## Example

The following example shows how to use the command to collect, in Windows, log information in the specified folder (C:\Users\folder01):

```
hcmds64getlogs /dir "C:\Users\folder01" /types Automation /arc AO_log
```

---

## Related topics

- [1.3 Valid characters for arguments in a command](#)
- 

## 1.7.6 restoresystem (restoring the JP1/AO system)

### Description

This command restores the backup data, such as the configuration and database information of JP1/AO, obtained by the `backupsystem` command.

The following list shows the data restored when the command is executed:

- Tasks<sup>#</sup>
- Debug tasks<sup>#</sup>
- Task histories
- Services
- Debug services
- Service templates
- Users
- User groups
- Service groups
- Connection destination definitions
- Shared service properties
- Various definition files

#

The status of restored tasks and debug tasks is changed after restoration as follows.

Table 1-14: Status of tasks and debug tasks at backup time and after restoration

Status of tasks and debug tasks at backup time	Status of tasks and debug tasks after restoration
Waiting	Canceled (The end date and time of the tasks and debug tasks are set to the date and time of restoration.)
Holding	
In Progress	Failed
Waiting for Response	
Abnormal Detection	

Status of tasks and debug tasks at backup time	Status of tasks and debug tasks after restoration
Terminated	Failed
Completed	Completed
Failed	Failed
Canceled	Canceled

## Syntax

```
restoresystem
  /dir backup-data-path
  [/auto]
```

## Arguments

*/dir backup-data-path*

This option specifies the absolute or relative path to the backup folder that stores the backup data specified in the `backsystem` command.

*/auto*

This option causes the command to automatically start and stop the services and databases of JP1/AO, JP1/OA, and the Hitachi Command Suite products. If this option is omitted, the services and databases of JP1/AO, JP1/OA, and the Hitachi Command Suite products are not automatically started and stopped.

If you want to use this option in a cluster environment, services registered with the cluster software must be offline.

## Located in

In Windows:

*JP1/AO-installation-folder\bin*

In Linux:

*/opt/jplao/bin*

## Execute permission

Execute the command as a user with Administrator or root permissions. If a user without Administrator or root permissions executes the command, a message appears asking the user to elevate the permission level.

## Remarks

- Executing this command creates a temporary file. For this reason, make sure that the folder in which backup files are stored has a sufficient free space. The required free space is as follows:  
Total size of the files to be backed up + 20 MB
- This command does not restore the files below. Manually set the following files again if necessary:
  - SSL server certificate file for HTTPS connection
  - Private key file for HTTPS connection
  - Private key file for public key authentication
  - Cluster service control command that was created (in a Linux cluster configuration)

Place the files for HTTPS connection in a location defined in the `user_httpsd.conf` file, and place the file for public key authentication in a location defined in the user-specified properties file (`config_user.properties`).

- If you do not specify the `auto` option, then make sure that JP1/AO services are not running before executing this command.

If these services are still running, execute the `hcnds64srv` command with the `stop` option to stop the services.

- Be careful if the JP1/AO installation path includes half-width space characters. If there is a file or folder whose path is the same as the string before the first space character in that path, the `restoresystem` command will fail with return code 114. If this happens, move that file or folder to a different path, re-install JP1/AO, and then execute the `restoresystem` command again.

For example, assume that JP1/AO is installed in `C:\Program Files\HITACHI\JP1AO`. In this case, the `restoresystem` command will fail if there is a file or folder whose path is `C:\Program`.

- In the restored tasks and debug tasks, the following operations cannot be selected:
  - Retry the Task From the Failed Step
  - Retry the Task From the Step After the Failed Step
- In the restored tasks and debug tasks, the progress and status of the step are not displayed.

## Return code

The following table lists the return codes from the command.

Return code	Description
0	The command succeeded.
1	The argument is invalid.
2	The command execution has been interrupted.
3	The service status is invalid.
4	One of the other commands is running.
7	An invalid path is specified.
9	The specified path does not exist.
10	The specified path is not accessible.
14	You do not have permission to execute the command.
110	Performing the restoration failed.
111	Starting or stopping the service failed.
113	The backup file is invalid.
114	An access to the scheduler database failed.
255	The command execution has been interrupted due to an error other than the above.

## Example

The following example shows how to use the command to restore, in Windows, data in the specified backup folder (`C:\Users\Backup`):

```
restoresystem /dir C:\Users\Backup /auto
```



---

## Related topics

- [1.7.1 backupsystem \(backing up the JP1/AO system\)](#)
  - [1.3 Valid characters for arguments in a command](#)
  - [Topic User-specified properties file \(config\\_user.properties\) in the JP1/Automatic Operation Configuration Guide](#)
-

# 2

## APIs

This chapter describes the API functions provided by JP1/AO. The HTTP or HTTPS protocol can be used for communication with the API functions.

## 2.1 List of APIs

The following tables list and describe the APIs that can be used for JPI/AO.

Table 2-1: List of API functions for service template functionality

API name	Function	See
Acquisition of a list of service templates	Acquires a list of service templates registered in JPI/AO.	2.4.1 Acquisition of a list of service templates
Acquisition of information about a service template	Acquires information about the specified service template.	2.4.2 Acquisition of information about a service template
Deletion of a service template	Deletes the specified service template.	2.4.3 Deletion of a service template
Acquisition of a list of operations for a service template	Acquires a list of operations that can be executed for the specified service template.	2.4.4 Acquisition of a list of operations for a service template
Acquisition of the HTML file necessary for importing a service template	Acquires the HTML file necessary for importing the specified service template.	2.4.5 Acquisition of the HTML file necessary for importing a service template
Import of a service template	Imports the specified service template.	2.4.6 Import of a service template
Acquisition of information necessary for exporting a service template	Acquires information necessary for exporting the specified service template.	2.4.7 Acquisition of information necessary for exporting a service template
Export of a service template	Exports the specified service template.	2.4.8 Export of a service template
Acquisition of the URL for displaying the details of a service template	Acquires the URL for displaying the details of the specified service template.	2.4.9 Acquisition of the URL for displaying the details of a service template
Acquisition of information necessary for creating a service based on a service template	Acquires information necessary for creating a service from the specified service template.	2.4.10 Acquisition of information necessary for creating a service based on a service template
Creation of a service based on a service template	Creates a service from the specified service template. You can specify properties when creating a service.	2.4.11 Creation of a service based on a service template

Table 2-2: List of APIs for service functionality

API name	Function	See
Acquisition of a list of services	Acquires a list of services registered in JPI/AO.	2.5.1 Acquisition of a list of services
Acquisition of service information	Acquires information about the specified service.	2.5.2 Acquisition of service information
Editing a service	Edits the specified service.	2.5.3 Editing a service
Deletion of a service	Deletes the specified service.	2.5.4 Deletion of a service
Acquisition of a list of operations for a service	Acquires a list of operations that can be executed for the specified service.	2.5.5 Acquisition of a list of operations for a service
Acquisition of information necessary for executing a service	Acquires information necessary for executing the specified service.	2.5.6 Acquisition of information necessary for executing a service
Execution of a service	Executes the specified service.	2.5.7 Execution of a service
Acquisition of information necessary for resetting the counter for a service	Acquires information necessary for resetting the counter for the specified service (initialization of statistics).	2.5.8 Acquisition of information necessary for resetting the counter for a service

API name	Function	See
Reset of the counter for a service	Resets the counter for the specified service (initialization of statistics).	<a href="#">2.5.9 Reset of the counter for a service</a>
Acquisition of information necessary for the operation to change the status of a service to release	Acquires information necessary for the operation to change the status of the specified service to release.	<a href="#">2.5.10 Acquisition of information necessary for the operation to change the status of a service to release</a>
Change of the status of a service to release	Changes the status of the specified service to release.	<a href="#">2.5.11 Change of the status of a service to release</a>
Acquisition of information necessary for the operation to change the status of a service to maintenance	Acquires information necessary for the operation to change the status of the specified service to maintenance.	<a href="#">2.5.12 Acquisition of information necessary for the operation to change the status of a service to maintenance</a>
Change of the status of a service to maintenance	Changes the status of the specified service to maintenance.	<a href="#">2.5.13 Change of the status of a service to maintenance</a>
Acquisition of information necessary for the operation to change the status of a service to disabled	Acquires information necessary for the operation to change the status of the specified service to disabled.	<a href="#">2.5.14 Acquisition of information necessary for the operation to change the status of a service to disabled</a>
Change of the status of a service to disabled	Changes the status of the specified service to disabled.	<a href="#">2.5.15 Change of the status of a service to disabled</a>
Acquisition of the URL for the details of a service	Acquires the URL for displaying the details of the specified service.	<a href="#">2.5.16 Acquisition of the URL for the details of a service</a>
Acquisition of information necessary for changing the version of the service template used by a service	Acquires information necessary for changing the version of the service template used by the specified service.	<a href="#">2.5.17 Acquisition of information necessary for changing the version of the service template used by a service</a>
Change of the version of the service template used by a service	Applies the service template of any version to the specified service.	<a href="#">2.5.18 Change of the version of the service template used by a service</a>

Table 2-3: List of APIs for schedule functionality

API name	Function	See
Acquisition of a list of schedules	Acquires a list of schedules set for a task.	<a href="#">2.6.1 Acquisition of a list of schedules</a>
Acquisition of schedule information	Acquires information about the specified schedule.	<a href="#">2.6.2 Acquisition of schedule information</a>
Acquisition of a list of operations for a schedule	Acquires a list of operations that can be executed for the specified schedule.	<a href="#">2.6.3 Acquisition of a list of operations for a schedule</a>
Acquisition of information necessary for canceling a schedule	Acquires information necessary for canceling the specified schedule.	<a href="#">2.6.4 Acquisition of information necessary for canceling a schedule</a>
Cancellation of a schedule	Cancels the specified schedule.	<a href="#">2.6.5 Cancellation of a schedule</a>
Acquisition of information necessary for pausing a schedule	Acquires information necessary for pausing the specified schedule.	<a href="#">2.6.6 Acquisition of information necessary for pausing a schedule</a>
Pause of a schedule	Pauses the specified schedule.	<a href="#">2.6.7 Pause of a schedule</a>

API name	Function	See
Acquisition of information necessary for resuming a schedule	Acquires information necessary for resuming the specified schedule.	<a href="#">2.6.8 Acquisition of information necessary for resuming a schedule</a>
Resume of a schedule	Resumes the specified schedule.	<a href="#">2.6.9 Resume of a schedule</a>

Table 2-4: List of APIs for task functionality

API name	Function	See
Acquisition of a list of tasks	Acquires a list of tasks.	<a href="#">2.7.1 Acquisition of a list of tasks</a>
Acquisition of task information	Acquires information about the specified task.	<a href="#">2.7.2 Acquisition of task information</a>
Editing a task	Edits the notes and TODO for the specified task.	<a href="#">2.7.3 Editing a task</a>
Deletion of a task	Deletes the specified task. If the specified task is not a debug task, this API function acquires the URL for archiving the task.	<a href="#">2.7.4 Deletion of a task</a>
Acquisition of a list of task operations	Acquires a list of operations that can be executed for the specified task.	<a href="#">2.7.5 Acquisition of a list of task operations</a>
Acquisition of information necessary for stopping task execution	Acquires information necessary for stopping execution of the specified task.	<a href="#">2.7.6 Acquisition of information necessary for stopping task execution</a>
Stoppage of task execution	Stops execution of the specified task.	<a href="#">2.7.7 Stoppage of task execution</a>
Acquisition of information necessary for forcibly stopping a task	Acquires information necessary for forcibly stopping the specified task.	<a href="#">2.7.8 Acquisition of information necessary for forcibly stopping a task</a>
Forced stoppage of a task	Forcibly stops the specified task.	<a href="#">2.7.9 Forced stoppage of a task</a>
Acquisition of information necessary for re-executing a task	Acquires information necessary for re-executing the specified task.	<a href="#">2.7.10 Acquisition of information necessary for re-executing a task</a>
Re-execution of a task	Re-executes the specified task.	<a href="#">2.7.11 Re-execution of a task</a>
Acquisition of information necessary for responding to a task that is in the status Waiting for Response	Acquires information necessary for responding to a task that is in the status Waiting for Response. Among the steps of the task that has the specified ID, information about the step that was least recently placed in the status Waiting for Response is acquired.	<a href="#">2.7.12 Acquisition of information necessary for responding to a task that is in the status Waiting for Response</a>
Response to a task that is in the status Waiting for Response	Among the steps of the task that has specified ID, performs a response input for the step that was least recently placed in the status Waiting for Response.	<a href="#">2.7.13 Response to a task that is in the status Waiting for Response</a>
Acquisition of information necessary for retrying a task (retry from the failed step)	Specifies a task, and acquires information necessary for retrying the task from the failed step.	<a href="#">2.7.14 Acquisition of information necessary for retrying a task (retry from the failed step)</a>
Retry from the failed step	Specifies a task, and retries the task from the failed step.	<a href="#">2.7.15 Retry from the failed step</a>

API name	Function	See
Acquisition of information necessary for retrying a task (retry from the step after the failed step)	Specifies a task, and acquires information necessary for retrying the task from the step after the failed step.	<a href="#">2.7.16 Acquisition of information necessary for retrying a task (retry from the step after the failed step)</a>
Retry from the step after the failed step	Specifies a task, and retries the task from the step after the failed step.	<a href="#">2.7.17 Retry from the step after the failed step</a>
Acquisition of information necessary for archiving a task	Acquires the argument template necessary for archiving the specified task.	<a href="#">2.7.18 Acquisition of information necessary for archiving a task</a>
Archiving a task	Archives the specified task.	<a href="#">2.7.19 Archiving a task</a>
Acquisition of a list of steps	Among the steps included in the specified task, acquires a list of steps displayed in the <b>Task Details</b> window.	<a href="#">2.7.20 Acquisition of a list of steps</a>
Acquisition of task logs	Acquires the task logs for the specified task.	<a href="#">2.7.21 Acquisition of task logs</a>

Table 2-5: List of history-related API functions

API name	Function	See
Acquisition of a list of history records	Acquires a list of history records.	<a href="#">2.8.1 Acquisition of a list of history records</a>
Deletion of history records (with conditions specified)	Deletes history records according to the conditions specified by query parameters.	<a href="#">2.8.2 Deletion of history records (with conditions specified)</a>
Acquisition of a history record	Acquires the history record that has the specified ID.	<a href="#">2.8.3 Acquisition of a history record</a>
Deletion of history records (with an ID specified)	Deletes the history record that has the specified ID.	<a href="#">2.8.4 Deletion of history records (with an ID specified)</a>
Acquisition of a list of operations for a history record	Acquires a list of operations that can be executed for the history record that has the specified ID.	<a href="#">2.8.5 Acquisition of a list of operations for a history record</a>

Table 2-6: List of property-related APIs

API name	Function	See
Acquisition of a list of property definitions	Acquires a list of property definitions.	<a href="#">2.9.1 Acquisition of a list of property definitions</a>
Acquisition of property definition information	Acquires information about the specified property definition.	<a href="#">2.9.2 Acquisition of property definition information</a>
Acquisition of a list of operations for a property definition	Acquires a list of operations that can be executed for the specified property definition.	<a href="#">2.9.3 Acquisition of a list of operations for a property definition</a>
Acquisition of lists of property definitions and property values	Acquires lists of property definitions and property values.	<a href="#">2.9.4 Acquisition of lists of property definitions and property values</a>
Acquisition of a list of property values	Acquires a list of the values of the following properties: <ul style="list-style-type: none"> <li>• Service share properties</li> <li>• Properties related to specific services</li> <li>• Properties related to specific schedules</li> </ul>	<a href="#">2.9.5 Acquisition of a list of property values</a>

API name	Function	See
Batch update of property values	Updates the following property values in a batch: <ul style="list-style-type: none"> <li>Property values related to specific tasks</li> <li>Property values related to specific services</li> <li>Service share property values</li> <li>Property values for multiple services</li> </ul>	<a href="#">2.9.6 Batch update of property values</a>
Acquisition of a property value	Acquires information about the specified property value.	<a href="#">2.9.7 Acquisition of a property value</a>
Update of a property value	Updates the property value that has the specified ID.	<a href="#">2.9.8 Update of a property value</a>
Acquisition of a list of operations for a property value	Acquires a list of operations for the specified property value.	<a href="#">2.9.9 Acquisition of a list of operations for a property value</a>
Acquisition of a list of property groups	Acquires a list of property groups that the properties retained by a service belong to.	<a href="#">2.9.10 Acquisition of a list of property groups</a>

Table 2-7: List of service group-related API functions

API name	Function	See
Acquisition of a list of service groups	Acquires a list of service groups.	<a href="#">2.10.1 Acquisition of a list of service groups</a>
Acquisition of information about a service group	Acquires information about the specified service group.	<a href="#">2.10.2 Acquisition of information about a service group</a>
Acquisition of a list of operations for a service group	Acquires a list of operations that can be executed for the specified service group.	<a href="#">2.10.3 Acquisition of a list of operations for a service group</a>

Table 2-8: List of tag-related API functions

API name	Function	See
Acquisition of a list of tag groups	Acquires a list of tag groups. In addition, this API function acquires a list of tags that belong to each tag group.	<a href="#">2.11.1 Acquisition of a list of tag groups</a>
Acquisition of a list of tags	Acquires a list of tags that are set for the specified resource.	<a href="#">2.11.2 Acquisition of a list of tags</a>

Table 2-9: List of APIs for information management

API name	Function	See
Acquisition of user information	Acquires information about users that execute API functions.	<a href="#">2.12.1 Acquisition of user information</a>
Acquisition of version information	Acquires the JP1/AO and API versions.	<a href="#">2.12.2 Acquisition of version information</a>

## 2.2 Specifications common to APIs

---

The following shows the specifications common to all APIs. Note that the API functions provided by JP1/AO follow the REST (Representational State Transfer) architecture style.

This section describes the specifications that are specific to JP1/AO. The specifications conform to HTTP1.1 unless otherwise described.

Note that *API* in this section refers to the *API provided by JP1/AO*, and user programs that use APIs (such as a portal program) are generally called *API clients*.

### 2.2.1 Communication protocol

The following shows the communication protocols and port numbers that are used by APIs.

- Communication protocol

API functions support the HTTP and HTTPS protocols. API functions use a protocol that is used by JP1/AO to communicate with a web browser. For both protocols, version 1.1 is supported. For the detailed specifications of the communication protocols, see the following standards:

- For the HTTP protocol:  
RFC2616
- For the HTTPS protocol:  
RFC2818

- Port number

The default port number setting differs depending on the communication protocol and the OS of the JP1/AO server.

- When the communication protocol is HTTP:  
22015
- When the communication protocol is HTTPS:  
22016

If you want to change the port number, see the topic *Procedure to change the port number* in the *JP1/Automatic Operation Configuration Guide*.

### 2.2.2 Security and authentication

User authentication is required to issue an API request and receive the response. A JP1/AO API uses the Basic authentication (Basic Access Authentication) or an authentication using the HSSO token. The HSSO token is necessary for Single Sign-On. The HSSO token is timed out when 1,000 seconds have passed since it was issued.

In the request header, specify the authentication information to be used for user authentication. The following example specifies authentication information in the request header.

#### Example

For Basic authentication:

```
Authorization: Basic c3lzdGVtOm1hbmFnZXI=
```



For authentication using the HSSO token

```
Authorization:HSSO 32bd25936120d68dceabcb49493079f8ef82a4_V0300
```

If a request with no permission is issued, the JP1/AO server returns status code 401 as the response, and requests user authentication.

### Tip

If Basic authentication or HSSO token-based authentication is used to connect to the JP1/AO server, `WWW-Authenticate: HSSO hssso token` is returned in the response header. If you want to use the same session to connect to the JP1/AO server and issue an API function, specify the request header as follows:

```
Authorization:HSSO hssso-token
```

## 2.2.3 Input/output format

The JSON format or XML format can be used as the data format for API request and response. Specify this data format in the request header. If you omit specifying the data format, the JSON format is set. UTF-8 is used as the character encoding for input/output format.

The following example specifies the request header when the XML format is specified as the input/output format.

### Example

```
Accept:application/xml  
Content-Type:application/xml
```

## 2.2.4 Namespace

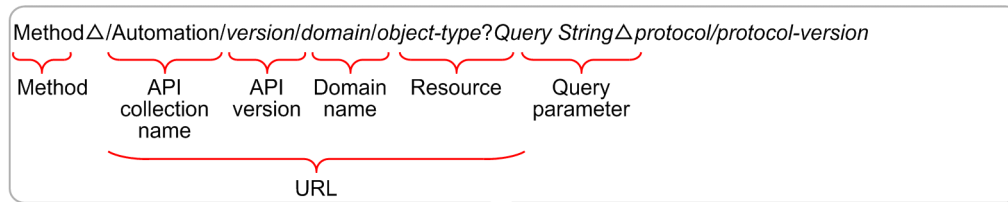
If the XML format is used as the data format for API request and response, use the following namespace:

- `http://www.hitachi.com/products/it/software/xml/restfw/common/API-version`
- `http://www.hitachi.com/products/it/software/xml/automation/API-version`

## 2.2.5 Request format

The following shows the request format required for the API to use the functions provided by JP1/AO.

Figure 2-1: Request format (an example when the domain is objects)



MethodΔ/Automation/version/domain/object-type?Query StringΔprotocol/protocol-version

Host: *host-name-or-IP-address:port-number*  
 Accept: *data-format*  
 Accept-Language: *language-code* (such as ja, zh, or en)  
 User-Agent: *software-information-about-the-API-client*

} Request header

Legend:  
 Δ: Half-width space

The following table describes the components of the request format.

Table 2-10: Components of the request format

Item	Description	See	
Method	Specify an operation for the resource.	<a href="#">2.2.7 Supported methods</a>	
URL	API collection name	API collection <sup>#</sup> name. Specify Automation as the fixed value.	--
	API version	Specify the API version to be used.	See API version in the description of each API.
	Domain name	Specify the domain name of the resource that you want to operate by executing the API function. This request format is used when the domain is <code>objects</code> .	<a href="#">2.2.8 Domain names and resources that can be managed by APIs</a>
	Resource	The functions provided by JP1/AO are provided as API resources. Specify a resource according to the processing you want to execute.	
Query parameter	By adding search conditions to the request, you can filter and sort output results in the response.	<a href="#">2.2.9 Query parameter</a>	
Protocol	Specify HTTP as the communication protocol used by the API. Specify HTTP even when you use HTTP as the communication protocol.	<a href="#">2.2.1 Communication protocol</a>	
Protocol version	Specify 1.1 as the version of the communication protocol used by the API.		
Request header	Host	Specify the host information.	<a href="#">2.2.10 Request header</a>
	Accept	Specify the data format of the response.	
	Accept-Language	Specify the language code for the response.	
	User-Agent	Specify the software information of the API client.	

Legend:  
 --: Not applicable.

#  
 A *collection* refers to data subject to processing.

For details and components of the request format, see Request format in the description of each API (differs depending on the API).

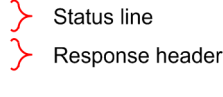
## 2.2.6 Response format

The following shows the response format.

Figure 2-2: Response format

```

protocol/protocol-version status-code message
Content-Type: data-format
Response body
  
```



The diagram shows the response format with two red curly brackets on the right side. The top bracket spans the line 'protocol/protocol-version status-code message' and is labeled 'Status line'. The bottom bracket spans the line 'Content-Type: data-format' and is labeled 'Response header'. The line 'Response body' is not bracketed.

The following table describes the components of the response format for a request.

Table 2-11: Components of the response format

Item		Description	See
Status line	Protocol	Displays the communication protocol used by the API.	--
	Protocol version	Displays the version of the communication protocol used by the API.	--
	Status code	Execution result of the request is returned as the status code.	<ul style="list-style-type: none"> <li>• See Status code in the description for each API.</li> <li>• For details about the status code when an error occurs before an API is executed, see the relevant topic in <a href="#">2.2.17 Status code</a>.</li> </ul>
	Message	Displays the contents of the status code.	
Response header	Content-Type	The response data format specified in the request header is returned.	<a href="#">2.2.13 Response header</a>
Response body		A schema of the data format specified in the request header is returned.	See Response schema in the description for each API.

Legend:

--: Not applicable.

## 2.2.7 Supported methods

In an API, an operation for a resource is defined as a method.

Specify a method according to the API processing. For details about the method to be specified, see Request format in the description of each API.

The following table describes the methods supported by an API.

Table 2-12: Supported methods

Method	Description
GET	Acquires the information and list of resources.
POST	Executes JP1/AO processing according to the resource.

## 2.2.8 Domain names and resources that can be managed by APIs

Specify the domain name for the resource to be operated by the API to be executed, and a resource supported by JP1/AO.

Note that, for XML requests and response data, the resource names for the `objects` domain are replaced with singular names.

The following table describes the list of domain names and resources that can be managed by APIs.

Table 2-13: Domain names and resources that can be managed by APIs

Domain name	Resource	Description of the resource
objects	ServiceTemplates	Service templates
	Services	Services registered in JP1/AO
	Schedules	Schedules set for tasks
	Tasks	Tasks created by execution of services
	FlowSteps	Steps included in a task
	TaskLogs	Task logs
	PropertyDefinitions	Definitions of service template properties or service share properties
	PropertyInformations	Property definitions and property values
	PropertyValues	Values of service properties, schedule properties, task properties, and service share properties
	PropertyGroups	Property groups
	ServiceGroups	Service groups
	TagGroups	Tag groups
	Tags	Tags
user	UserInfo <sup>#</sup>	Information about a user authenticated by JP1/AO
configuration	VersionInfo <sup>#</sup>	Information about JP1/AO and API versions

#

This resource name is not specified for a request because it is included in a response. For details about how to specify a request, see Request format in the description of each API.

## 2.2.9 Query parameter

By using query parameters to add search conditions to a request, you can filter and sort output results in the response.

This section describes query parameters supported by JP1/AO.

### Query parameter that can be specified for all APIs

The query parameter described in the following table can be specified for all APIs.

Table 2-14: Query parameter that can be specified for all APIs

Parameter	Description	Specifiable value	Default value
alt	Input/output data format can be specified in the same way as the Content-Type header and Accept header in a request.	xml or json	--

Legend:

--: Not applicable.

## Query parameters that can be used for some APIs

The query parameters in the table below can be specified for a part of APIs<sup>#</sup>.

#

- Acquisition of a list of service templates
- Acquisition of a list of services
- Acquisition of a list of schedules
- Acquisition of a list of tasks
- Acquisition of a list of steps
- Acquisition of task logs
- Acquisition of a list of history records
- Acquisition of a list of property definitions
- Acquisition of a list of property values
- Acquisition of lists of property definitions and property values
- Acquisition of a list of property groups
- Acquisition of a list of service groups
- Acquisition of a list of tag groups
- Acquisition of a list of tags

Table 2-15: Query parameters that can be specified for a part of APIs

Parameter	Description	Specifiable value <sup>#1</sup>	Default value
HQL::filter	Filters the output results by using the specified conditions.	See 2.2.11 Using HQL standard.	--
HQL::fields	Specify this parameter when you want to filter members to be included in a response. You can specify multiple parameters by separating them by commas (.).	<i>Member-name</i>	--
HQL::sortBy	Sorts the output results by the specified member name.	<i>Member-name</i> [{ASC DESC}](, <i>member-name</i> [{ASC DESC}]) <ul style="list-style-type: none"> <li>• ASC: Ascending order</li> <li>• DESC: Descending order</li> </ul>	ASC
HQL::offset <sup>#2</sup>	Specifies the position of the heading object whose information is to be acquired. Specify the maximum number of objects that can be included in a response with HQL::count. <i>page</i> takes preference over HQL::offset.	0 to 2147483647	0

Parameter	Description	Specifiable value <sup>#1</sup>	Default value
HQL::count <sup>#2</sup>	Specify the maximum number of objects that can be included in a response, starting from the position of the heading object specified with HQL::offset. pageSize takes preference over HQL::count. If the total of HQL::count and HQL::offset exceeds 2,147,483,647, the objects of the position specified with HQL::offset until the 2,147,483,647th are acquired.	1 to 2147483647	100
page <sup>#2</sup>	Acquires information about the specified page when a resource is divided into pages. You must also specify pageSize. page takes preference over HQL::offset.	1 to 2147483647	--
pageSize <sup>#2</sup>	Specify the maximum number of objects that can be displayed in a page. pageSize takes preference over HQL::count.	1 to 2147483647	--

Legend:

--: Not applicable.

#1

If you want to specify a character string that cannot be expressed as a URL, use UTF-8 encoding and encode the character string.

#2

You cannot specify the parameters for the following API functions: Acquisition of task logs, Acquisition of a list of steps, and Acquisition of a list of property groups.

For the pageSize parameter, specify the maximum number of objects to be displayed in a page. For the page parameter, specify the number of the page to be displayed from among the divided pages. They basically resemble the items **Rows/page** and **Page of Services of Services List** in a JP1/AO window. If you specify the parameters page and pageSize, the numbers of all resources and pages are returned to the Pagination object. From the value of this object, you can determine whether the next page exists.

Note that the parameters page and pageSize are used by converting to an HQL::offset value according to the following formula:

$$\text{HQL::offset} = \text{pageSize} * (\text{page} - 1)$$

Therefore, if the parameters page and pageSize exceed the range that can be specified for HQL::offset, the status code 400 (Bad Request) is returned.

---

## Related topics

- [2.2.11 Using HQL standard](#)
- 

## 2.2.10 Request header

The request header specifies the data format and language code for the response.

Table 2-16: Request header

Header	Description	Specifiable value	Default value	Whether specification is required
Host	Specify the following items as host information: <ul style="list-style-type: none"> <li>Host name or IP address: Host name or IP address of the JP1/AO server</li> <li>Port number: Port number that an API uses to connect to JP1/AO</li> </ul> When you specify the port number, see 2.2.1 Communication protocol.	Specify this value after checking the user environment.	--	Required
Accept	Specify the desired data format for response data.	<ul style="list-style-type: none"> <li>application/json: JSON format</li> <li>application/xml: XML format</li> <li>multipart/form-data: multipart format<sup>#1</sup></li> <li>text/html: HTML format<sup>#2</sup></li> </ul>	application/json	Required
Accept-Language	Specify the desired language code for response data.	<ul style="list-style-type: none"> <li>ja or ja-JP: Japanese</li> <li>zh or zh-CN: Chinese</li> <li>en or en-US: English</li> <li>Locale for other regions: English</li> </ul>	en	Required
Content-Type	Specify the data format for the request body.	<ul style="list-style-type: none"> <li>application/json: JSON format</li> <li>application/xml: XML format</li> <li>application/octet-stream: octet-stream format<sup>#3</sup></li> </ul>	application/json	Optional
Authorization	Specify authentication information.	For Basic authentication: <i>user-information</i>  For authentication using the HSSO token: <i>hssso-token</i>	--	Optional

Legend:

--: Not applicable.

#1

Valid only for the API function Import of a service template.

#2

Valid only for the API function Acquisition of the HTML file necessary for importing a service template.

#3

Valid only for the API function Export of a service template.

## 2.2.11 Using HQL standard

By specifying HQL (Hitachi Query Language) for HQL::filter, you can filter the target data. A *collection* refers to data to be filtered.

Use UTF-8 encoding and encode characters and symbols that cannot be expressed as a URL.

### Format

To define a collection of a resource request, use the following expressions:

```
expression ::= "(" expression ")" | binary-expression | expression junction
expression
junction ::= ( "and" | "or" )

binary-expression ::= ( compare-expression | tuple-expression )

compare-expression ::= name-expression compare-operation value-expression

name-expression ::= property-name | "[" property-name "]"
compare-operation ::= ( "eq" | "=" | "ne" | "<>" | "!=" | "gt" | ">" | "lt" | "<" |
"ge" | ">=" | "le" | "<=" | "starts" | "ends" )
value-expression ::= ( string-expression | number-expression | Boolean-expression )

string-expression ::= "'" ([^'] | [']{2})* "'"
number-expression ::= ( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" )+
Boolean-expression ::= "true" | "false" | "TRUE" | "FALSE"

tuple-expression ::= name-expression tuple-operation tuple-value-expression

tuple-value-expression ::= "[" value-expression ("," value-expression)* "]"

tuple-operation ::= ( "in" | "not in" )
```

*property-name* is a member name defined in a resource.

*value-expression* displays whether the expression is a string expression, number expression, or Boolean expression. This value is different from the actual data type of the member defined in a resource.

The following table describes the relationship between the data types and expression formats.

Table 2-17: Relationship between data types and expression formats

Data type	Expression format
integer/long	number-expression
enum	string-expression
string	string-expression
ISO8601String	string-expression
URLString	string-expression
boolean	Boolean-expression

The following table lists and describes the operators supported by HQL in preferential order.



Table 2-18: Operators supported by HQL

Operator	Description	Specifiable data type	Priority
eq	Equal	number-expression, string-expression (string, enum), Boolean-expression	1
ne	Not equal	number-expression, string-expression (string, enum), Boolean-expression	1
gt	Greater than	number-expression, string-expression (string, enum)	1
lt	Smaller than	number-expression, string-expression (string, enum)	1
ge	Equal or greater than	number-expression, string-expression (string, enum)	1
le	Equal or smaller than	number-expression, string-expression (string, enum)	1
starts <sup>#</sup>	Start value	string-expression (string, excluding ISO8601String)	1
ends <sup>#</sup>	End value	string-expression (string, excluding ISO8601String)	1
in	Included	number-expression, string-expression (string, enum), Boolean-expression	1
not in	Not included	number-expression, string-expression (string, enum), Boolean-expression	1
and	Both true	compare-expression, tuple-expression	2
or	Either of them true	compare-expression, tuple-expression	3

#

The operators are not case sensitive.

## Usage example

The example below filters the specified line. If you want to specify a character string that cannot be expressed as a URL, use UTF-8 encoding and encode the character string.

Before URL encoding:

```
...?HQL::filter=instanceID in [1000,1001,1002] and status = 'running'
```

After URL encoding:

```
...?HQL::filter=instanceID%20in%20%5b1000%2c1001%2c1002%5d%20and%20status%20%3d%20%27running%27
```

## 2.2.12 Domain object format

A *domain* refers to a location in which resources supported by JP1/AO are stored. A *domain object* refers to a resource. This section describes the data formats of the members that resources have.

## Supported data type

The following table describes the data types supported by the JSON format and XML format.

Table 2-19: Supported data type

Type name	Description
boolean	true or false
integer	32-bit signed integer
long	64-bit signed integer
string <sup>#</sup>	Text data

#

ISO8601String, URLString, and enum are string-type extended expressions.

## Date and time

The following describes how to specify the date and time for a domain object.

To specify the date and time, use ISO8601 format. In this format, you can omit all information except year (yyyy). If the date or time is omitted, the minimum specifiable value is automatically added. If the time zone is omitted, the time zone set for the JP1/AO server is set by default.

Note, however, that you cannot omit the date and time if you use HQL::filter to specify them. If you acquire time information in JSON format, the time is output in a format where a colon (:) is not used for time zone information (for example, 2014-12-09T18:50:30.500+0900). To specify the time information acquired in JSON format as an input for an API, add a colon (:) in the time zone (for example, 2014-12-09T18:50:30.500+09:00). If you do not add a colon (:), an error occurs.

Note that a year, month, date, time, and time zone are displayed in the response body in the format yyyy-mmddThh:mm:ss.mmmTZD if the data type of a resource member is ISO8601String.

Table 2-20: Format of year-month-date, time, and time zone

Format	Example	Time processed by JP1/AO
yyyy-mm-ddThh:mm:ss.mmmTZD	2014-12-09T18:50:30.500+09:00	Same as the example.
yyyy-mm-ddThh:mm:ss.mmm	2014-12-09T18:50:30.500.000	2014-12-09T18:50:30.500.000[time-zone-of-the-host-server]
yyyy-mm-ddThh:mm:ssTZD	2014-12-09T18:50:30+09:00	2014-12-09T18:50:30.000+09:00
yyyy-mm-ddThh:mmTZD	2014-12-09T18:50+09:00	2014-12-09T18:50:00.000+09:00
yyyy-mm-ddThhTZD	2014-12-09T18+09:00	2014-12-09T18:00:00.000+09:00
yyyy-mm-dd	2014-12-09	2014-12-09T00:00:00.000[time-zone-of-the-host-server]
yyyy-mm	2014-12	2014-12-01T00:00:00.000[time-zone-of-the-host-server]
yyyy	2014	2014-01-01T00:00:00.000[time-zone-of-the-host-server]

## 2.2.13 Response header

The following table describes the response headers controlled by JP1/AO.

Table 2-21: Response headers

Header	Description
Cache-Control	Performs cache control on the response information of an API for which the GET method is specified.
Content-Type	Data format of the response data
Content-disposition	Added to indicate that the response data is an attachment.
Language	Language code of the response data
Location	URL information. This information is different from the URL information for the request. This header displays the URL information for redirection if you must acquire the response data.
WWW-Authenticate	Outputs the authenticated HSSO token.
Warning	Displays information when the API processing succeeds but there is a problem with the status of the server.

## Related topics

- [2.2.10 Request header](#)

## 2.2.14 Members of resources

Functions provided by JP1/AO are categorized into resources. In the response body, you can acquire resource information as members. The function-based table below shows the name, data type, description, and whether HQL::filter and HQL::sortBy is applied, for each returned resource member.

For details on how to specify a year, month, and date, see [Table 2-20: Format of year-month-date, time, and time zone](#), unless otherwise described.

Table 2-22: Members that can be acquired by Acquisition of a list of services (Resource (ServiceTemplate))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
keyName	string	Service template ID	Y
displayName	string	Display name of the service template	Y
iconURL	URLString	URL of the icon image that is set for the service template	N
vendorID	string	Vendor ID	Y
version	string	Version of the service template	Y
vendorName	string	Vendor name	Y
tags	string	List of tags added to the service template	N
createTime	ISO8601String	Year, month, date, time, and time zone at which the service template was created	Y

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
modifyTime	ISO8601String	Year, month, date, time, and time zone at which the service template was updated	Y
description	string	Description of the service template	Y
releaseState	enum	Release state of the service template <ul style="list-style-type: none"> <li>• debug: Debug</li> <li>• release: Release</li> </ul>	Y
latest	boolean	Whether the service template is the latest version <ul style="list-style-type: none"> <li>• true: Yes</li> <li>• false: No</li> </ul>	Y
imageUrl	URLString	URL of the service overview image	N
supportedScheduleType	enum csv	Schedule type that can be applied to the service template <ul style="list-style-type: none"> <li>• immediate: Executed immediately.</li> <li>• schedule: Executed at the specified date and time.</li> <li>• recurrence: Executed periodically.</li> </ul>	Y
needVUP	boolean	Whether there is a service using a service template of an older version <ul style="list-style-type: none"> <li>• true: Yes</li> <li>• false: No</li> </ul>	Y
componentOutdated	boolean	Whether the service template contains a component of an older version <ul style="list-style-type: none"> <li>• true: Yes</li> <li>• false: No</li> </ul>	Y
usedServices	integer	Number of services using the service template	N
usedTemplates	integer	Number of service templates using the service template as a service component	N
disableFeatures	string	Invalid operation for the service template	Y
supportedActionType#	string	Operations that can be performed for the task: <ul style="list-style-type: none"> <li>• forciblyStop: Forcibly stop the task</li> <li>• retry: Retry the task</li> </ul>	Y
pagination	Object	Information when the resource is divided into pages	N
page	integer	page specified in the request (page number)	N
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

#

If supportedActionType is not specified, all operations are permitted.

Table 2-23: Members that can be acquired by Acquisition of a list of services (Resource (Services))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
name	string	Service name	Y
description	string	Description of the service	Y
tags	string	Tag information separated by commas ( , ) into tag units is displayed.	N
serviceTemplateName	string	Name of the service template that is used as the base of a service	Y
createTime	ISO8601String	Year, month, date, time, and time zone at which the service was created	Y
modifyTime	ISO8601String	Year, month, date, time, and time zone at which the service information was updated	Y
serviceState	enum	Service type <ul style="list-style-type: none"> <li>• debug: Debug</li> <li>• test: Test</li> <li>• release: Release</li> <li>• maintenance: Maintenance</li> <li>• disabled: Disabled</li> </ul>	Y
serviceGroupName	string	Name of the service group that the service belongs to	Y <sup>#</sup>
iconURL	URLString	URL of the icon image that is set for the service template	N
vendorName	string	Vendor name for the service template that is used as the base of the service	Y
version	string	Version of the service template that is used as the base of the service	Y
lastSubmitTime	ISO8601String	Year, month, date, time, and time zone at which the service was last executed by the user	Y
favorite	boolean	Whether the service is registered as a favorite <ul style="list-style-type: none"> <li>• true: Yes</li> <li>• false: No</li> </ul>	Y
failedCount	integer	Number of times tasks that were generated from the service failed	Y
completedCount	integer	Number of times tasks that were generated from the service ended normally	Y
lastFailedTime	ISO8601String	Year, month, date, time, and time zone at which a task that was generated from the service last failed	Y

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
resetTime	ISO8601String	Year, month, date, time, and time zone at which the counter for the service was last reset.	Y
executedCount	integer	Number of times tasks that were generated from the service were executed	Y
latest	boolean	Whether the service template used by the service is the latest version <ul style="list-style-type: none"> <li>• true: Yes</li> <li>• false: No</li> </ul>	Y
imageUrl	URLString	URL of the service overview image	N
supportedScheduleType	enum csv	Schedule type of the service <ul style="list-style-type: none"> <li>• immediate: Executed immediately.</li> <li>• schedule: Executed at the specified date and time.</li> <li>• recurrence: Executed periodically.</li> </ul>	Y
submitCount	integer	Number of times the service was executed	Y
serviceTemplateID	long	ID of the service template that is used as the base of the service	Y
serviceGroupID	long	Service group ID	Y
supportedActionType	string	Operations that can be performed for the task: <ul style="list-style-type: none"> <li>• forciblyStop: Forcibly stop the task</li> <li>• retry: Retry the task</li> </ul>	Y
pagination	Object	Information used when a resource is divided into pages	N
page	integer	The page specified in the request (page number)	N
pageSize	integer	The page size specified in the request (maximum number of objects included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	The number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

#

When All Resources is specified, the service is treated as if DefaultServiceGroup is specified.

Table 2-24: Members that can be acquired by Acquisition of a list of schedules (Resource (Schedules))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
name	string	Task name specified when the service is executed	Y
submitter	string	Execution user name	Y
status	enum	Status of the schedule of the periodic execution task <ul style="list-style-type: none"> <li>complete: The schedule of the periodic execution task is complete. The task will not be executed.</li> <li>running: The schedule of the periodic execution task is running. The task will be executed periodically.</li> </ul>	Y
scheduleType	enum	Schedule type <ul style="list-style-type: none"> <li>immediate: Executed immediately</li> <li>schedule: Executed at the specified date and time.</li> <li>recurrence: Executed periodically.</li> </ul>	Y
createTime	ISO8601String	Year, month, date, time, and time zone at which the schedule was created by service execution	Y
modifyTime	ISO8601String	Year, month, date, time, and time zone at which task information was updated	Y
description	string	Description of the task	Y
scheduledStartTime	ISO8601String	Year, month, date, time, and time zone at which scheduled task is planned to start	Y <sup>#</sup>
recurrenceInterval	enum	Periodic execution cycle <ul style="list-style-type: none"> <li>daily: Daily</li> <li>weekly: Weekly</li> <li>monthly: Monthly</li> </ul>	Y
recurrenceMinutes	integer	Interval (minutes) of the service is to be executed when the periodic execution cycle is set to daily: <ul style="list-style-type: none"> <li>60</li> <li>120</li> <li>180</li> <li>240</li> <li>360</li> <li>480</li> <li>720</li> <li>1440</li> </ul>	N
recurrenceDayOfWeek	string	Day of week the service is to be executed when the periodic execution cycle is set to Weekly (1: Sun to 7: Sat)	N
recurrenceDayOfMonth	string	Day of month the service is to be executed when the periodic execution cycle is set to Monthly (1st to 31st)	N
recurrenceLastDayOfMonth	boolean	Whether to execute on the final day of month <ul style="list-style-type: none"> <li>true: Execute</li> </ul>	Y

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
recurrenceLastDayOfMonth	boolean	<ul style="list-style-type: none"> <li>false: Not execute</li> </ul>	Y
recurrenceStartDate	string	Date the periodic execution task starts execution (yyyy-mm-dd)	Y
recurrenceTime	string	Time the periodic execution task is executed (hh:mm:ss)	Y
serviceState	enum	Service type <ul style="list-style-type: none"> <li>debug: Debug</li> <li>test: Test</li> <li>release: Release</li> <li>maintenance: Maintenance</li> </ul>	Y
serviceID	long	ID of the service that is the generation source of the schedule	Y
supportedActionType	string	Operations that can be performed for the task: <ul style="list-style-type: none"> <li>forciblyStop: Forcibly stop the task</li> <li>retry: Retry the task</li> </ul>	Y
pagination	Object	Information when the resource is divided into pages	N
page	integer	The page number specified in the request (page number)	N
pageSize	integer	The page size specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

#

HQL::sortBy is not applied.

Table 2-25: Members that can be acquired by Acquisition of a list of tasks (Resource (Tasks))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
name	string	Task name	Y
status	enum	Status of the task	Y



Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
status	enum	<ul style="list-style-type: none"> <li>failed: Failed</li> <li>completed: Completed</li> <li>canceled: Canceled</li> <li>inProgressTerminating: Being stopped</li> <li>inProgressWithError: Abnormality detected</li> <li>waitingForInput: Waiting for response</li> <li>inProgress: In progress</li> <li>suspended: Suspended</li> <li>waiting: Waiting</li> <li>longRunning: Long running</li> </ul>	Y
startTime	ISO8601String	Start year, month, date, time, and time zone of the task	Y
completionTime	ISO8601String	End year, month, date, time, and time zone of the task	Y
scheduledStartTime	ISO8601String	Year, month, date, time, and time zone at which the scheduled task is planned to be started	Y
submitter	string	Execution user name	Y
submitTime	ISO8601String	Year, month, date, time, and time zone at which the task was created by service execution	Y
modifyTime	ISO8601String	Year, month, date, time, and time zone at which the task information was updated	Y
serviceState	enum	Task type <ul style="list-style-type: none"> <li>debug: Debug</li> <li>test: Test</li> <li>release: Release</li> <li>maintenance: Maintenance</li> <li>buildDebug: Executed from the debugger.</li> </ul>	Y
scheduleType	enum	Schedule type <ul style="list-style-type: none"> <li>immediate: Executed immediately.</li> <li>schedule: Executed at the specified date and time.</li> <li>recurrence: Executed periodically.</li> </ul>	Y
description	string	Description of the task	Y
serviceName	string	Name of the service that is the generation source of the task	Y
tags	string	List of tags that are added to the task	Y
recurrenceInterval	enum	Execution interval of the periodic execution task <ul style="list-style-type: none"> <li>daily: Daily</li> <li>weekly: Weekly</li> <li>monthly: Monthly</li> </ul>	Y
recurrenceTime	string	Execution time of the periodic execution task (hh:mm:ss)	Y
recurrenceStartDate	ISO8601String	Start year, month, and date of the periodic execution task (yyyy-mm-dd)	Y
serviceGroupName	string	Name of the service group that the generation source service of the task belongs to	Y
toDo	boolean	Whether TODO is set for the task	Y

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
toDo	boolean	<ul style="list-style-type: none"> <li>true: Yes</li> <li>true: No</li> </ul>	Y
notes	string	Notes added to the task	Y
stepStartTime	long	Year, month, date, time, and time zone at which a step included in the task was executed for the first time	Y
serviceTemplateID	long	ID of the service template that is used as the base of the task	Y
scheduleID	long	ID of the schedule that is used as the base of the task	Y
serviceGroupID	long	ID of the service group that the generation source service of the task belongs to	Y
serviceID	long	ID of the service that is the generation source of the task	Y
pagination	Object	Information when the resource is divided into pages	N
page	integer	The page specified in the request (page number)	N
pageSize	integer	The page size specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

Table 2-26: Members that can be acquired by Response to a task that is in the status Waiting for Response (Resource (ResponseInput))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
instanceID	string	Instance ID	N
dialogText	string	Character string displayed in the <b>Add Response</b> window	N
labelButton0	string	Option 0	N
labelButton1	string	Option 1	N
labelButton2	string	Option 2	N
labelButton3	string	Option 3	N
labelButton4	string	Option 4	N
labelButton5	string	Option 5	N
labelButton6	string	Option 6	N

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
labelButton7	string	Option 7	N
labelButton8	string	Option 8	N
labelButton9	string	Option 9	N
screenURL	string	URL for displaying the <b>Add Response</b> window	N
pagination	Object	Information when the resource is divided into pages	N
page	integer	page specified in the request (page number)	N
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

N: Not applied.

Table 2-27: Members that can be acquired by Acquisition of a list of steps (Resource (FlowSteps))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	string	Instance ID	N
name	string	Step name	N
startTime	string	Start year, month, date, and time of the step (yyyy-MM-dd hh:mm:ss)	N
completionTime	string	End year, month, date, and time of the step (yyyy-MM-dd hh:mm:ss)	N
jobStatus	enum	Status of the step <ul style="list-style-type: none"> <li>• noplan: Not scheduled</li> <li>• normal: Normal</li> <li>• warning: Warning</li> <li>• waiting: Waiting</li> <li>• holding: Being held</li> <li>• break: Interrupted</li> <li>• break_after: Interrupted (After Execution)</li> <li>• running: Running</li> <li>• waiting_for_response: Waiting for response</li> <li>• abnormal_continue: Abnormality detected</li> <li>• complete: Completed</li> <li>• error: Failed</li> <li>• abnormal: Ended with a warning</li> </ul>	N

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
jobStatus	enum	<ul style="list-style-type: none"> <li>un_exec: Not executed but ended</li> <li>bypass: Bypassed and not executed</li> <li>terminate: Terminated</li> <li>waiting_for_foreach: Waiting to repeat</li> </ul>	N
comment	string	Comment for the step	N
stepStatus	enum	Status of the step (JP1/AO) <ul style="list-style-type: none"> <li>normal: Normal</li> <li>warning: Warning</li> <li>waiting: Waiting</li> <li>holding: Being held</li> <li>break: Interrupted</li> <li>break_after: Interrupted (After Execution)</li> <li>running: Running</li> <li>waiting_for_response: Waiting for response</li> <li>abnormal_continue: Abnormality detected</li> <li>complete: Completed</li> <li>error: Failed</li> <li>abnormal: Ended with a warning</li> <li>un_exec: Not executed but ended</li> <li>bypass: Bypassed and not executed</li> <li>terminate: Terminated</li> <li>waiting_for_foreach: Waiting to repeat</li> </ul>	N
pagination	Object	Information when a resource is divided into pages	N
page	integer	The page specified in the request (page number)	N
pageSize	integer	The page size specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

N: Not applied.

Table 2-28: Members that can be acquired by Acquisition of task logs (Resource (Tasklogs))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	N
text	string	Body of the task log	N

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
totalSize	long	Total file size of the task logs (unit: byte)	N
readSize	long	Size of the acquired task log (unit: byte)	N
lineCount	long	Number of lines in the acquired task log	N
offset	long	Offset specified when the task log is acquired (unit: byte)	N
reverse	boolean	Whether the task log was acquired in the opposite direction from the offset <ul style="list-style-type: none"> <li>true: The task log was acquired in the opposite direction from the offset.</li> <li>false: The task log was acquired in the normal direction from the offset.</li> </ul>	N
pagination	Object	Information when a resource is divided into pages	N
page	integer	page specified in the request (page number)	N
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

N: Not applied.

Table 2-29: Members that can be acquired by Acquisition of a list of history records (Resource (TaskHistory))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
name	string	Task name	Y
submitter	string	Execution user name	Y
serviceName	string	Name of the generation source service of the task	Y
tags	string	Tag information (CSV format)	N
scheduleType	enum	Schedule type <ul style="list-style-type: none"> <li>immediate: Executed immediately.</li> <li>schedule: Executed at the specified date and time.</li> <li>recurrence: Executed periodically.</li> </ul>	Y
scheduledStartTime	ISO8601String	Start year, month, date, time, and time zone of the task that is executed at the specified date and time	Y

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
startTime	ISO8601String	Start year, month, date, time, and time zone of the task	Y
completionTime	ISO8601String	End year, month, date, time, and time zone of the task	Y
stepStartTime	ISO8601String	Start year, month, date, time, and time zone of the long-running task	Y
recurrenceInterval	enum	Execution interval of the periodic execution task <ul style="list-style-type: none"> <li>daily: Daily</li> <li>weekly: Weekly</li> <li>monthly: Monthly</li> </ul>	Y
recurrenceMinutes	integer	Interval (minutes) of the service is to be executed when the periodic execution cycle is set to daily: <ul style="list-style-type: none"> <li>60</li> <li>120</li> <li>180</li> <li>240</li> <li>360</li> <li>480</li> <li>720</li> <li>1440</li> </ul>	N
recurrenceDayOfWeek	string	(When the periodic execution interval is weekly) Day of the week when the service is executed (1: Sunday to 7: Saturday)	N
recurrenceDayOfMonth	string	(When the periodic execution interval is monthly) Day when the service is executed (1 to 31)	N
recurrenceLastDayOfMonth	boolean	Whether to execute the service on the last day of each month <ul style="list-style-type: none"> <li>true: Yes</li> <li>false: No</li> </ul>	Y
recurrenceTime	string	Execution time of the periodic execution task (hh:mm:ss)	Y
archiveTime	ISO8601String	Year, month, date, time, and time zone at which the task was archived	Y
taskID	long	Task ID	Y
submitTime	ISO8601String	Year, month, date, time, and time zone at which the task was executed	Y
recurrenceStartDate	ISO8601String	Start date of the periodic execution task (yyyy-mm-dd)	Y
status	enum	Status of the task <ul style="list-style-type: none"> <li>failed: Failed</li> <li>completed: Ended normally</li> <li>canceled: Canceled</li> <li>inProgressTerminating: Being stopped</li> <li>inProgressWithError: Abnormality detected</li> <li>waitingForInput: Waiting for response</li> <li>inProgress: In progress</li> <li>suspended: Suspended</li> <li>waiting: Waiting</li> <li>longRunning: Long running</li> </ul>	Y
description	string	Description of the task	Y

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
serviceState	enum	Release status of the service <ul style="list-style-type: none"> <li>• debug: Debug</li> <li>• test: Test</li> <li>• release: Release</li> <li>• maintenance: Maintenance</li> </ul>	Y
toDo	boolean	Whether TODO is set for the task <ul style="list-style-type: none"> <li>• true: Yes</li> <li>• false: No</li> </ul>	Y
notes	string	Notes added to the task	Y
serviceGroupName	string	Name of the service group that the generation source service of the history record belongs to	Y
serviceGroupID	long	ID of the service group that the generation source service of the history record belongs to	Y
pagination	Object	Information when the resource is divided into pages	N
page	integer	page specified in the request (page number)	N
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

Table 2-30: Members that can be acquired by Acquisition of a list of property definitions (Resource (PropertyDefinitions))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
keyName	string	Property key name	Y
displayName	string	Display name of the property	N
defaultValue	string	Default value of the property	Y
type	enum	Data type of the property <ul style="list-style-type: none"> <li>• boolean</li> </ul>	Y

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
type	enum	<ul style="list-style-type: none"> <li>integer</li> <li>string</li> <li>double</li> <li>timestamp(date)</li> <li>password</li> <li>list</li> <li>file</li> </ul>	Y
visibility	enum	Visibility of the property <ul style="list-style-type: none"> <li>config</li> <li>exec</li> </ul>	Y
scope	enum	Valid range of the property <ul style="list-style-type: none"> <li>share: Service share property</li> <li>local: Property that is valid only for the service</li> </ul>	Y
description	string	Description of the property	N
mode	enum	Input/output type of the property <ul style="list-style-type: none"> <li>in: Input property</li> <li>out: Output property</li> </ul>	Y
required	boolean	Whether the property must be specified to execute the service <ul style="list-style-type: none"> <li>true: Required.</li> <li>false: Can be omitted.</li> </ul>	Y
maxLength	integer	Maximum length of a character string that can be input in the property	Y
minLength	integer	Minimum length of a character string that can be input in the property	Y
minValue	string	Minimum value that can be input in the property	Y
maxValue	string	Maximum value that can be input in the property	Y
pattern	string	Regular expression pattern of a character string that can be specified for the property <code>string</code> or <code>password</code>	Y
valueList	string	Candidate property values that are separated by commas (,) when the data type of the property is list	Y
propertyGroupName	string	Property group name	Y
validationScript	string	Property validation processing JavaScript	Y
readOnly	boolean	Whether to suppress a change of the property value <ul style="list-style-type: none"> <li>true: Yes</li> <li>false: No</li> </ul>	Y
hidden	boolean	Whether to suppress a display of the property <ul style="list-style-type: none"> <li>true: Yes</li> <li>false: No</li> </ul>	Y
reference	boolean	Whether the property value references another property value <ul style="list-style-type: none"> <li>true: Yes</li> <li>false: No</li> </ul>	Y
serviceTemplateID	long	Service template ID	Y



Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
pagination	Object	Information when the resource is divided into pages	N
page	integer	page specified in the request (page number)	N
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

Table 2-31: Members that can be acquired by Acquisition of lists of property definitions and property values (Resource (PropertyInformation))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
keyName	string	Property key name	Y
displayName	string	Display name of the property	N
defaultValue	string	Default value of the property	Y
value	string	Current value of the property	Y
type	enum	Data type of the property <ul style="list-style-type: none"> <li>boolean</li> <li>integer</li> <li>string</li> <li>double</li> <li>timestamp</li> <li>password</li> <li>list</li> <li>file</li> </ul>	Y
visibility	enum	Visibility of the property <ul style="list-style-type: none"> <li>config: Displayed as an input item for the <b>Service Definition</b> window.</li> <li>exec: Displayed as an input item for the <b>Service Definition</b> window and the <b>Submit Service</b> window</li> </ul>	Y
scope	enum	Valid range of the property <ul style="list-style-type: none"> <li>share: Service share property</li> </ul>	Y

Member name	Data type	Description	Whether HQL::filter or HQL::sort By is applied
scope	enum	<ul style="list-style-type: none"> <li>local: Property that is valid only for the service</li> </ul>	Y
description	string	Description of the property	N
mode	enum	Input/output type of the property <ul style="list-style-type: none"> <li>in: Input property</li> <li>out: Output property</li> </ul>	Y
required	boolean	Whether the property must be specified to execute the service <ul style="list-style-type: none"> <li>true: Required.</li> <li>false: Can be omitted.</li> </ul>	Y
maxLength	integer	Maximum length of a character string that can be input in the property	Y
minLength	integer	Minimum length of a character string that can be input in the property	Y
minValue	string	Minimum value that can be input in the property	Y
maxValue	string	Maximum value that can be input in the property	Y
pattern	string	Regular expression pattern of a character string that can be specified for the property <code>string</code> or <code>password</code>	Y
valueList	string	Candidate property values that are separated by commas (,) when the data type of the property is list	Y
propertyGroupName	string	Property group name	Y
validationScript	string	Property validation processing JavaScript	N
readOnly	boolean	Whether to suppress a change of the property value <ul style="list-style-type: none"> <li>true: Yes</li> <li>false: No</li> </ul>	Y
hidden	boolean	Whether to suppress a display of the property <ul style="list-style-type: none"> <li>true: Yes</li> <li>false: No</li> </ul>	Y
reference	boolean	Whether the property value refers another property value <ul style="list-style-type: none"> <li>true: Yes</li> <li>false: No</li> </ul>	N
serviceTemplateID	long	Service template ID for the resource	Y
serviceID	long	Service ID for the resource	Y
taskID	long	Task ID for the resource	Y
scheduleID	long	Schedule ID for the resource	Y
pagination	Object	Information when the resource is divided into pages	N
page	integer	page specified in the request (page number)	N
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N

Member name	Data type	Description	Whether HQL::filter or HQL::sort By is applied
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

Table 2-32: Members that can be acquired by Acquisition of a list of property values (Resource (PropertyValues))

Member name	Data type	Description	Whether HQL::filter or HQL::sort By is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
type	enum	Data type of the property <ul style="list-style-type: none"> <li>• boolean</li> <li>• integer</li> <li>• string</li> <li>• double</li> <li>• timestamp(date)</li> <li>• password</li> <li>• list</li> <li>• file</li> </ul>	Y
keyName	string	Property key name	Y
value	string	Property value	Y
serviceID	long	Service ID of the resource	Y
scheduleID	long	Schedule ID of the resource	Y
taskID	long	Task ID of the resource	Y
pagination	Object	Information when the resource is divided into pages	N
page	integer	page specified in the request (page number)	N
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

Table 2-33: Members that can be acquired by Acquisition of a list of property groups (Resource (PropertyGroup))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
keyName	string	Property group ID	Y
displayName	string	Display name of the property group	N
description	string	Description of the property group	N
ordinal	integer	Display sequence number of the property group	N
validationScript	string	JavaScript for validation processing between properties in the property group	N
display	enum	Whether to display the property group <ul style="list-style-type: none"> <li>submit: Displayed in the <b>Submit Service</b> window.</li> <li>config: Displayed in the <b>Edit Service</b> window.</li> <li>taskDetail: Displayed in the <b>Task Details</b> window.</li> </ul>	N
pagination	Object	Information when the resource is divided into pages	N
page	integer	page specified in the request (page number)	N
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Table 2-34: Members that can be acquired by Acquisition of a list of service groups (Resource (ServiceGroup))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
objectID	string	Instance ID for the Resource Group resource	Y
name	string	Name of the service group	Y
description	string	Description of the service group	Y
pagination	Object	Information when the resource is divided into pages	N
page	integer	page specified in the request (page number)	N

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

Table 2-35: Members that can be acquired by Acquisition of a list of tag groups (Resource (TagGroup))

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
name	string	Name of the tag group	Y
tag	string	List of tags that belong to the tag group (CSV format)	N
pagination	Object	Information when the resource is divided into pages	N
page	integer	page specified in the request (page number)	N
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

Legend:

Y: Applied. N: Not applied.

Table 2-36: Members that can be acquired by Acquisition of a list of tags (Resource (Tag)) (when the detail query parameter is not specified)

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
name	string	Tag name	Y
tagGroupID	long	ID of the tag group that the tag belongs to	Y
pagination	Object	Information when the resource is divided into pages	N
page	integer	page specified in the request (page number)	N
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

Table 2-37: Members that can be acquired by Acquisition of a list of tags (Resource (Tag)) (when the detail query parameter is specified)

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
data	Object	List of resources	N
instanceID	long	Instance ID	Y
name	string	Tag name	Y
usedTemplates	integer	Number of release and development service templates using the tag	Y
usedServices	integer	Number of services using the tag	Y
usedTasks	integer	Number of tasks using the tag	Y
usedHistories	integer	Number of history records using the tag	Y
usedPlugins	integer	Number of release plug-ins using the tag	Y
usedDevelopPlugins	integer	Number of development plug-ins using the tag	Y
usedDevelopTemplates	integer	Number of development service templates using the tag	Y

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
tagGroupID	long	ID of the tag group that the tag belongs to	Y
pagination	Object	Information when the resource is divided into pages	N
page	integer	page specified in the request (page number)	N
pageSize	integer	pageSize specified in the request (maximum number of objects that can be included in a page)	N
numPages	integer	Total number of pages (page number)	N
totalCount	integer	Total number of returned resources	N
count	integer	Number of data items that match the conditions specified by query parameters (0 to n)	N

Legend:

Y: Applied. N: Not applied.

Table 2-38: Members of a resource for information management functionality (UserInfo)

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
userName	string	User name	N
accessPermission	string	Access permissions granted to the user	N
fullName	string	Full name of the user	N
description	string	Description of the user	N
email	string	Email address for the user	N
resourceGroup	ResourceGroup	Access permissions granted to the user for each Resource Group resource	N

Legend:

N: Not applied.

Table 2-39: Members of a resource for information management functionality (ResourceGroup)

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
instanceID	string	Instance ID	N
name	string	Name of the Resource Group resource	N
description	string	Description of the Resource Group resource	N

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
accessPermission	string[]	Access permissions granted to the user for each Resource Group resource	N

Legend:

N: Not applied.

Table 2-40: Members of a resource for information management functionality (VersionInfo)

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
productName	string	Product name (JP1/Automatic Operation)	N
productVersion	string	Product version	N
apiVersion1	string	API version	N

Legend:

N: Not applied.

Table 2-41: Members of a resource for information management functionality (Information)

Member name	Data type	Description	Whether HQL::filter or HQL::sortBy is applied
message	string	Message	N
messageID	string	Message ID	N

Legend:

N: Not applied.

## 2.2.15 Members to be returned for APIs that execute JP1/AO operations

Some API functions provided by JP1/AO execute JP1/AO operations. Applicable API functions and returned members are shown below. For details about requests, see the request format in the description of each API function.

### APIs that execute JP1/AO operations

- Import of a service template
- Export of a service template
- Creation of a service based on a service template



- Execution of a service
- Reset of the counter for a service
- Change of the status of a service to `release`
- Change of the status of a service to `maintenance`
- Change of the status of a service to `disabled`
- Change of the version of the service template used by a service
- Cancellation of a schedule
- Pause of a schedule
- Resume of a schedule
- Stoppage of task execution
- Forced stoppage of a task
- Re-execution of a task
- Response to a task that is in the status `Waiting for Response`
- Retry from the failed step
- Retry from the step after the failed step
- Archiving a task

Table 2-42: Members to be returned for APIs that execute JP1/AO operations

Member name	Data type	Description
<code>instanceID</code>	string	Indicates the instance ID.
<code>created</code>	string	Indicates (in ISO8601 format) the date and time the object was generated.
<code>updated</code>	string	Indicates the time this object was updated, if asynchronous processing was executed. If synchronous processing is executed, this member indicates the same time as <i>created</i> . This member is expressed in ISO8601 format.
<code>completed</code>	string	Indicates the time the processing was completed, if asynchronous processing was executed. If synchronous processing is executed, this member indicates the same time as <i>created</i> . This member is expressed in ISO8601 format.
<code>state</code>	string	<ul style="list-style-type: none"> <li>• <code>queued</code>: The operation has not started processing yet. In this status, only stop processing is accepted.</li> <li>• <code>running</code>: The operation is running. In this status, only stop processing is accepted.</li> <li>• <code>failed</code>: The operation failed.</li> <li>• <code>success</code>: The operation has been successfully completed.</li> <li>• <code>stopping</code>: The operation is being stopped.</li> <li>• <code>stopped</code>: The operation is stopped before completion.</li> </ul>
<code>affectedResource</code>	string	Indicates the URL of the API resource created or updated as a result of the operation.

## 2.2.16 Members to be returned for API functions that acquire executable operations

Some API functions provided by JP1/AO acquire operations that can be executed for resources, and execution-destination URLs. Applicable API functions and returned members are shown below. For details about requests, see the request format in the description of each API function.

## APIs that acquire executable operations

- Acquisition of a list of operations for a service template
- Acquisition of the HTML file necessary for importing a service template
- Acquisition of information necessary for exporting a service template
- Acquisition of the URL for displaying the details of a service template
- Acquisition of information necessary for creating a service based on a service template
- Acquisition of a list of operations for a service
- Acquisition of information necessary for executing a service
- Acquisition of information necessary for resetting the counter for a service
- Acquisition of information necessary for the operation to change the status of a service to `release`
- Acquisition of information necessary for the operation to change the status of a service to `maintenance`
- Acquisition of information necessary for the operation to change the status of a service to `disabled`
- Acquisition of the URL for the details of a service
- Acquisition of information necessary for changing the version of the service template used by a service
- Acquisition of a list of operations for a schedule
- Acquisition of information necessary for canceling a schedule
- Acquisition of information necessary for pausing a schedule
- Acquisition of information necessary for resuming a schedule
- Deletion of a task
- Acquisition of a list of task operations
- Acquisition of information necessary for stopping task execution
- Acquisition of information necessary for forcibly stopping a task
- Acquisition of information necessary for re-executing a task
- Acquisition of information necessary for responding to a task that is in the status `Waiting for Response`
- Acquisition of information necessary for retrying a task (retry from the failed step)
- Acquisition of information necessary for retrying a task (retry from the step after the failed step)
- Acquisition of information necessary for archiving a task
- Acquisition of a list of operations for a history record
- Acquisition of a list of operations for a property definition
- Acquisition of a list of operations for a property value
- Acquisition of a list of operations for a service group

Table 2-43: Members to be returned for APIs that acquire executable operations

Member name	Data type	Description
name	string	Operation name
href	string	Execution-destination URL of the operation
method	string	Method name <ul style="list-style-type: none"><li>• GET</li></ul>

Member name	Data type	Description
method	string	• POST
parameters	Object	Parameters required when operations are executed

## 2.2.17 Status code

The following table describes the various status codes that can be returned when an API is executed. The status codes to be returned depend on the API, so see the description for each API for details.

Table 2-44: Status code

Status code	Message	Description
200	OK	Processing the request has been successfully completed.
201	Created	If creation of a resource ended successfully, status 201 is returned instead of status 200.
400	Bad Request	The content of the request is invalid.
401	Unauthorized	Authentication failed. Authentication information or permission information is invalid. The accepted authentication method is reported by the WWW-Authenticate response header. Specify the accepted authentication method in the Authorization request header. Alternatively, the user does not have a permission for the service group or the User Management permission.
403	Forbidden	The user does not have execution permission for the request.
404	Not found	The requested resource does not exist, or the user does not have permission to operate the requested resource. Alternatively, a specified query parameter is invalid.
405	Method not allowed	The requested method does not exist for this resource.
406	Not acceptable	The format of the specified response is not supported.
409	Conflict	The request cannot be completed because the data conflicts with data that already exists on the server, or because the system cannot accept the request in the current status.
412	Precondition failed	The request cannot be accepted because it does not satisfy requirements.
415	Unsupported media type	The format of the specified request is not supported.
500	Server-side error	A server processing error occurred.

### Related topics

- [2.2.18 Error information](#)

## 2.2.18 Error information

This section describes the case an error occurs in an API request. If an error occurs in an API request, the schema in the table below is returned as the response information. For error information other than the schema in the table below, see the manual *JPI/Automatic Operation Messages*.

The following table describes the schema of error information.

Table 2-45: Schema of error information

Member name	Data type	Description
errorSource	string	API where the error occurred
message	string	Message of the error
messageID	string	Message ID. If the error was caused by a wrong XML description in a request, <code>generic error</code> is set for the message ID.
application	string	Information about the application that holds the API where an error occurred (Automation)
messageData	string	Detailed information of the error

## Output example

The following example outputs KNAE02102-E as messageID of the error information.

```
{
  "errorSource" : "http://10.196.184.238:22015/Automation/v1/objects/Tasks/555",
  "message" : "The specified resource does not exist or you do not have access.
After reviewing the content of the following, please re-run.\n- The presence or
absence of resources\n- Access rights to the resource",
  "messageID" : "KNAE02102-E",
  "application" : "Automation"
}
```

## 2.3 API description format

---

The items below provide descriptions for individual APIs. Note that some items might not be described for some APIs.

### Function

Describes the function of an API.

### Execution permissions

Indicates the permissions and roles that are required to execute an API.

### API version

Indicates the version of an API.

### Request format

Describes the request format for requesting the use of an API.

### Status code

Describes the status code after you execute an API by using the HTTP or HTTPS protocol. For details about the status code when an error occurs before an API is executed, see [2.2.17 Status code](#).

### Response schema

Describes schema information of the response that is returned when an API is successfully completed.

### Usage example

Provides examples of the request for the use of API, and its response.

Note that the HTTP protocol is used in the examples. If the HTTPS protocol is used, replace *HTTP* with *HTTPS* when you read the description.

## 2.4 Service template-related API functions

### 2.4.1 Acquisition of a list of service templates

#### Function

Acquires a list of service templates registered in JP1/AO.

#### Execution permissions

Admin role, Develop role, Modify role

#### API version

v1

#### Request format

```
GET http://host:port/Automation/version/objects/ServiceTemplates
```

This API function acquires a list of all service templates for which the user who executed the API function has permissions. By specifying query parameters, you can filter the service templates for which you want to acquire the list. Specify query parameters in the following format:

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-46: List of query parameters that can be specified for the API function Acquisition of a list of service templates

Query parameter	Filter condition
tags	Whether all values are contained. You can specify multiple values by separating them with a comma ( , ).
q	For the following schema, a full-text search is performed to determine whether the specified value is contained: <ul style="list-style-type: none"><li>• keyName</li><li>• displayName</li><li>• vendorID</li><li>• vendorName</li><li>• tags</li><li>• description</li></ul> If you specify multiple values by separating them with a half-width space character, a full-text search is performed to determine whether all of the specified values are contained. This query parameter is not case sensitive.
usingServiceTemplateID	Service component containing the specified values
vendorID	Equal to the specified value. The query parameters are not case sensitive.
keyName	
version	

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have permission to acquire service templates.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

Data that matches the conditions specified by query parameters in a request is returned in the response body. The following shows the schema of the response body.

```
{
  "data" : [ {"member-of-the-resources-for-service-template-
functionality(ServiceTemplates)" : value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API function acquires a list of all service templates.

Request header:

```
GET /Automation/v1/objects/ServiceTemplates HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 00:34:32 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
998ebb201belcf76e7491a1380c4c54d5a59b7_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
```

```

"data" : [ {
  "instanceID" : 560,
  "keyName" : "remoteCommandExe",
  "displayName" : "Execute Remote Command",
  "iconURL" : "http://10.196.184.182:22015/Automation/icon/services/
com.hitachi.software.dna.cts/remoteCommandExe/01.20.00",
  "vendorID" : "Hitachi,Ltd.",
  "version" : "01.20.00",
  "vendorName" : "Hitachi,Ltd.",
  "tags" : "Execute Script,Linux,Windows",
  "createTime" : "2015-07-29T15:27:02.000+09:00",
  "modifyTime" : "2015-07-29T15:27:02.000+09:00",
  "description" : "Executes a command on the remote execution target server.",
  "releaseState" : "release",
  "latest" : true,
  "supportedScheduleType" : "immediate,schedule,recurrence",
  "needVUP" : false,
  "componentOutdated" : false,
  "usedServices" : 0,
  "usedTemplates" : 0,
  "supportedActionType" : "forciblyStop,retry"
}, {
  "instanceID" : 1116,
  "keyName" : "SP_GenericApplication",
  "displayName" : "Allocate Volumes for Generic Application",
  "iconURL" : "http://10.196.184.182:22015/Automation/icon/services/
com.hitachi.software.dna.cts/SP_GenericApplication/01.20.00",
  "vendorID" : "Hitachi,Ltd.",
  "version" : "01.20.00",
  "vendorName" : "Hitachi, Ltd.",
  "tags" : "Add New Storage",
  "createTime" : "2015-07-29T16:48:25.000+09:00",
  "modifyTime" : "2015-07-29T16:48:25.000+09:00",
  "description" : "Intelligent allocation service that uses sets of volumes from
the associated infrastructure group to be consumed by server(s) running a generic
application",
  "releaseState" : "release",
  "latest" : true,
  "imageURL" : "http://10.196.184.182:22015/Automation/services/custom/
000000000001116/SP_GenericApplication_overview.png",
  "supportedScheduleType" : "immediate,schedule",
  "needVUP" : false,
  "componentOutdated" : false,
  "usedServices" : 0,
  "usedTemplates" : 0,
  "supportedActionType" : "forciblyStop,retry"
} ],
"count" : 2
}

```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.4.2 Acquisition of information about a service template

### Function

Acquires information about the specified service template.



## Execution permissions

Admin role, Develop role, Modify role

## API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/ServiceTemplates/id
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have permission to acquire service templates.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : instance-id,
  "keyName" : "key-name",
  "displayName" : "display-name",
  "iconURL" : "icon-URL",
  "vendorID" : "vendor-ID",
  "version" : "version",
  "vendorName" : "vendor-name",
  "tags" : "tag",
  "createTime" : "created-date-and-time",
  "modifyTime" : "updated-date-and-time",
  "description" : "description",
  "releaseState" : "release-state",
  "latest" : {true|false},
  "imageURL" : "imageURL",
  "supportedScheduleType" : "supported-schedule-type",
  "needVUP" : {true|false},
  "componentOutdated" : {true|false},
  "usedServices" : used-services,
  "usedTemplates" : used-Templates,
  "disableFeatures" : "disable-features",
  "supportedActionType" : "supported-action-type"
}
```

## Usage example

In the following example, the API function acquires information about the service template whose instanceID is 1116.

Request header:

```
GET /Automation/v1/objects/ServiceTemplates/1116 HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 00:36:51 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
a9a6768131e2eff3ecbd5e4457f49e82e0506c_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : 1116,
  "keyName" : "SP_GenericApplication",
  "displayName" : "Allocate Volumes for Generic Application",
  "iconURL" : "http://10.196.184.182:22015/Automation/icon/services/
com.hitachi.software.dna.cts/SP_GenericApplication/01.20.00",
  "vendorID" : "com.hitachi.software.dna.cts",
  "version" : "01.20.00",
  "vendorName" : "Hitachi, Ltd.",
  "tags" : "Add New Storage",
  "createTime" : "2015-07-29T16:48:25.000+09:00",
  "modifyTime" : "2015-07-29T16:48:25.000+09:00",
  "description" : "Intelligent allocation service that uses sets of volumes from the
associated infrastructure group to be consumed by server(s) running a generic
application",
  "releaseState" : "release",
  "latest" : true,
  "imageURL" : "http://10.196.184.182:22015/Automation/services/custom/
000000000001116/SP_GenericApplication_overview.png",
  "supportedScheduleType" : "immediate,schedule",
  "needVUP" : false,
  "componentOutdated" : false,
  "usedServices" : 0,
  "usedTemplates" : 0,
  "supportedActionType" : "forciblyStop,retry"
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
-

## 2.4.3 Deletion of a service template

### Function

Deletes the specified service template.

### Execution permissions

Admin role, Develop role

### API version

v1

### Request format

```
DELETE http://host:port/Automation/version/objects/ServiceTemplates/id
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
204	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have permission to delete service templates.
409	Conflict	There is a service generated based on the specified service template, or there is a service template using the specified service template as a service plug-in.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Usage example

In the following example, the API function deletes the service template whose instanceID is 1116.

Request header:

```
DELETE /Automation/v1/objects/ServiceTemplates/1116 HTTP/1.1
Authorization: Basic c3lzZGVtOm1hbWFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 204 No Content
Date: Thu, 30 Jul 2015 00:39:20 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
7cfe7ffcd3e5603af8b08e3d2abdfafc5da41e3_V1o8Y30JdDBUB31jJSVPaRtjBSA=_V0810
```

```
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.4.4 Acquisition of a list of operations for a service template

### Function

Acquires a list of operations that can be executed for the specified service template.

### Execution permissions

Admin role, Develop role, Modyfy role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/ServiceTemplates/id/actions
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "data" : [ {
    "name" : "delete",
    "href" : "http://host:port/Automation/version/objects/ServiceTemplates/id",
    "method" : "DELETE",
    "parameters" : []
  }
]
```

```

    }, {
      "name" : "export",
      "href" : "http://host:port/Automation/version/objects/ServiceTemplates/id/
actions/export/invoke",
      "method" : "POST",
      "parameters" : []
    }, {
      "name" : "detailhelp",
      "href" : "http://host:port/Automation/version/objects/ServiceTemplates/id/
actions/detailhelp",
      "method" : "GET",
      "parameters" : []
    }, {
      "name" : "bind",
      "href" : "http://host:port/Automation/version/objects/ServiceTemplates/id/
actions/bind/invoke",
      "method" : "POST",
      "parameters" : []
    } ],
    "count" : count
  }
}

```

## Usage example

In the following example, the API function acquires a list of operations for the service template whose instanceID is 1116.

Request header:

```

GET /Automation/v1/objects/ServiceTemplates/1116/actions HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json

```

Response header:

```

HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 00:39:20 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
7cfe7ffcd3e5603af8b08e3d2abdfafc5da41e3_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json

```

Response body:

```

{
  "data" : [ {
    "name" : "delete",
    "href" : "http://10.196.184.182:22015/Automation/v1/objects/ServiceTemplates/
1116",
    "method" : "DELETE",
    "parameters" : [ ]
  }, {
    "name" : "export",
    "href" : "http://10.196.184.182:22015/Automation/v1/objects/ServiceTemplates/

```

```

1116/actions/export/invoke",
  "method" : "POST",
  "parameters" : [ ]
}, {
  "name" : "detailhelp",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/ServiceTemplates/
1116/actions/detailhelp",
  "method" : "GET",
  "parameters" : [ ]
}, {
  "name" : "bind",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/ServiceTemplates/
1116/actions/bind/invoke",
  "method" : "POST",
  "parameters" : [ ]
} ],
"count" : 4
}

```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.4.5 Acquisition of the HTML file necessary for importing a service template

### Function

Acquires the HTML file necessary for importing a service template. Note that authentication information is not added to the HTML file. Before executing the API function, make sure that you log in to JP1/AO to secure the session.

### Execution permissions

Admin role, Develop role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/services/ServiceTemplates/actions/import
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have import permission.

Status code	Message	Description
406	Not acceptable	The specified Accept header is invalid.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
<html>
<body>
<form method="POST" action="http://host:port/Automation/version/services/
ServiceTemplates/actions/import/invoke"
  enctype="multipart/form-data">
<input name="file" type="file"></input>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

## Usage example

In the following example, the API function acquires the HTML file necessary for importing a service template.

Request header:

```
GET /Automation/v1/services/ServiceTemplates/actions/import HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: text/html
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 00:40:59 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
ea15867727ce4f2cd07d5a48a3dedf919a34577_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: text/html
```

Response body:

```
<html>
<body>
<form method="POST" action="http://10.196.184.182:22015/Automation/v1/services/
ServiceTemplates/actions/import/invoke" enctype="multipart/form-data">
  <input name="file" type="file"></input>
  <input type="submit" value="Submit">
</form>
```

```
<body>
</html>
```

## Related topics

- [2.2.14 Members of resources](#)

## 2.4.6 Import of a service template

### Function

Imports the specified service template.

### Execution permissions

Admin role, Develop role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/services/ServiceTemplates/actions/import/
invoke
```

In the request body, specify a service template (.st or .zip).

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	The specified file is not a .st or .zip file. Alternatively, the specified .st or .zip file is corrupted or invalid.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have import permission.
412	Precondition failed	The server is not available.
415	Unsupported media type	The specified Content-Type header is invalid.
500	Server-side error	An attempt to store the temporary folder failed, or a server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.



```

{
  "instanceID" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ],
  "result" : [ {...} ],
  "resultType" : "result-type"
}

```

## Usage example

In the following example, the API function imports a service template (SP\_GenericApplication\_01.20.00.st).

Request header:

```

POST /Automation/v1/services/ServiceTemplates/actions/import/invoke HTTP/1.1
Authorization: Basic c3lzdGVtOm1hbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
Content-Length: 2106265
Expect: 100-continue
Content-Type: multipart/form-data; boundary=-----5564f06622f7727e

```

Response header:

```

HTTP/1.1 100 Continue
HTTP/1.1 200 OK
Date: Wed, 29 Jul 2015 07:48:21 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
9c9f012d1d34b9ede86d68728604c884b85e8_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json

```

Response body:

```

{
  "instanceID" : "f4c5065a-ff42-45df-bca9-e2d79b4b5bb7",
  "created" : "2015-07-29T16:48:26.528+09:00",
  "updated" : "2015-07-29T16:48:26.528+09:00",
  "completed" : "2015-07-29T16:48:26.528+09:00",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.182:22015/Automation/v1/objects/ServiceTemplates/1116" ],
  "result" : [ {
    "message" : "The service template was imported successfully (service template file name: SP_GenericApplication_01.20.00.st).",
    "messageID" : "KNAE03111-I"
  } ]
}

```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.4.7 Acquisition of information necessary for exporting a service template

### Function

Acquires information necessary for exporting the specified service template.

### Execution permissions

Admin role, Develop role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/ServiceTemplates/id/actions/export
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The user does not have permission to acquire service templates, or the service template does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "export",
  "href" : "http://host:port/Automation/version/objects/ServiceTemplates/id/actions/export/invoke",
  "method" : "POST",
  "parameters" : []
}
```

## Usage example

In the following example, the API function acquires information necessary for exporting the service template whose instanceID is 1116.

Request header:

```
GET /Automation/v1/objects/ServiceTemplates/1116/actions/export HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 00:42:05 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
c21cd879a4c62f90d8f7c5775ec1194e88a92b_v1o8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "export",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/ServiceTemplates/1116/
actions/export/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.4.8 Export of a service template

### Function

Exports the specified service template.

### Execution permissions

Admin role, Develop role

### API version

v1

## Request format

```
POST http://host:port/Automation/version/objects/ServiceTemplates/id/actions/export/
invoke
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
406	Not acceptable	The specified Accept header is invalid.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Usage example

In the following example, the API function exports the service template whose instanceID is 1116.

Request header:

```
POST /Automation/v1/objects/ServiceTemplates/1116/actions/export/invoke HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/octet-stream
Content-Type: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 01:58:34 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
866ad68aa7c23e457456b5b08479fb62250fdf_Vl08Y30JdDBUB31jJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Content-disposition: attachment;
filename="com.hitachi.software.dna.cts_SP_GenericApplication_01.20.00.st"
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/octet-stream
```

Response body:

```
Binary formatted "com.hitachi.software.dna.cts_SP_GenericApplication_01.20.00.st"
```

## Related topics

- [2.2.14 Members of resources](#)

## 2.4.9 Acquisition of the URL for displaying the details of a service template

### Function

Acquires the URL for displaying the details of the specified service template.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/ServiceTemplates/id/actions/detailhelp
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "export",
  "href" : "Link-to-the-detail-help",
  "method" : "POST",
  "parameters" : []
}
```

### Usage example

In the following example, the API function acquires the URL for displaying the details of the service template whose instanceID is 1116.

Request header:

```
GET /Automation/v1/objects/ServiceTemplates/1116/actions/detailhelp HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

```
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 02:04:35 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
11baaddb4ff5c120dlcca95c75fab1417d2c921_Vlo8Y30JdDBUB3ljJSVParTjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "detailhelp",
  "href" : "http://10.196.184.182:22015/Automation/services/custom/000000000001116/
r_all_vol_details.html",
  "method" : "GET",
  "parameters" : [ ]
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.4.10 Acquisition of information necessary for creating a service based on a service template

### Function

Acquires information necessary for creating a service based on the specified service template.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/ServiceTemplates/id/actions/bind
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The user does not have permission to acquire service templates, or the service template does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "bind",
  "href" : "http://host:port/Automation/version/objects/ServiceTemplates/id/actions/
bind/invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

## Usage example

In the following example, the API function acquires information necessary for creating a service based on the service template whose instanceID is 560.

Request header:

```
GET /Automation/v1/objects/ServiceTemplates/560/actions/bind HTTP/1.1
Authorization: Basic c3lzZGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 02:08:29 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
77efd47709df8b7f65468cb4778e804db1e6c_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "bind",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/ServiceTemplates/560/
actions/bind/invoke",
  "method" : "POST",
  "parameters" : [ {
```

```

    "name" : "Execute Remote Command",
    "description" : "Executes a command on the remote execution target server.",
    "tags" : "Execute Script, Linux, Windows",
    "serviceTemplateName" : "remoteCommandExe",
    "serviceState" : "test",
    "serviceGroupName" : "DefaultServiceGroup",
    "supportedScheduleType" : "immediate, schedule, recurrence",
    "serviceTemplateID" : 560
  }, {
    "type" : "string",
    "keyName" : "common.targetHost",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }, {
    "type" : "string",
    "keyName" : "common.remoteCommand",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }, {
    "type" : "string",
    "keyName" : "common.remoteCommandParameter",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  } ]
}

```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.4.11 Creation of a service based on a service template

### Function

Creates a service based on the specified service template. You can specify property values when creating a service.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/objects/ServiceTemplates/id/actions/bind/
invoke
```

The following shows the structure of the request body.

```
{
  "name" : "bind",
```



```

"href" : "http://host:port/Automation/version/objects/ServiceTemplates/id/actions/
bind/invoke",
"method" : "POST",
"parameters" : [ {...} ]
}

```

The following table describes the objects that can be specified as *parameters* (member) in the schema of a request.

Table 2-47: Objects that can be specified as parameters (member)

Function	Resource name	Number	Description
Service	Services	1	Service to be created
Property value	PropertyValues	0	Input property for the service

The following describes the properties that must be specified for the above objects.

Resource name	Member name	Number
Services	name	1
	description	
	tags	
	supportedScheduleType	
	serviceState	
	serviceGroupName	
PropertyValues	value	0 to n

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	Failed due to one of the following reasons: <ul style="list-style-type: none"> <li>The argument is invalid.</li> <li>Permissions allocated to the service group are invalid.</li> <li>The specified service name already exists.</li> <li>The number of services or tags has reached the maximum.</li> </ul>
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have permission to create services.
404	Not found	The user does not have permission to acquire service templates, or the service template does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "bind",
  "href" : "http://host:port/Automation/version/objects/ServiceTemplates/id/actions/
bind/invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

## Usage example

In the following example, the API function creates a service based on the service template whose instanceID is 560.

Request header:

```
POST /Automation/v1/objects/ServiceTemplates/560/actions/bind/invoke HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
Content-Type: application/json
Content-Length: 1001
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 02:30:37 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
fdef80b1cbd2d625cdbda39c16fda15f68a3d8c_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "55e8c5b7-b0ab-4016-ba62-f334b67c20c4",
  "created" : "2015-07-30T11:30:39.042+09:00",
  "updated" : "2015-07-30T11:30:39.042+09:00",
  "completed" : "2015-07-30T11:30:39.042+09:00",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.182:22015/Automation/v1/objects/Services/
2004" ],
  "result" : [ ]
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
-

## 2.5 Service-related APIs

This section describes the operations for managing service resources.

### 2.5.1 Acquisition of a list of services

#### Function

Acquires a list of services registered in JP1/AO.

#### Execution permissions

Admin role, Develop role, Modify role, Submit role

#### API version

v1

#### Request format

```
GET http://host:port/Automation/version/objects/Services
```

This API function acquires a list of all services for which the user who executed the API function has permissions. By specifying query parameters, you can filter the services for which you want to acquire the list. Specify query parameters in the following format:

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-48: List of query parameters that can be specified for the API function Acquisition of a list of services

Query parameter	Filter condition
serviceGroupID	Equal to the specified value
serviceTemplateID	
favorite	
propertyKey	keyName for a PropertyValues resource that contains the specified value
tags	Whether all values are contained. You can specify multiple values by separating them with a comma (,).
q	<p>For the following schema, a full-text search is performed to determine whether the specified value is contained:</p> <ul style="list-style-type: none"><li>• name</li><li>• description</li><li>• tags</li><li>• serviceTemplateName</li><li>• vendorName</li></ul> <p>If you specify multiple values by separating them with a half-width space character, a full-text search is performed to determine whether all of the specified values are contained. This query parameter is not case sensitive.</p>

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

Data that matches the conditions specified by query parameters in a request is returned in the response body. The following shows the schema of the response body.

```
{
  "data" : [ {"member-of-the-resources-for-service-functionality(Services)" :
value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API acquires a list of all services.

Request header:

```
GET /Automation/v1/objects/Services HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
Host: 10.196.184.182:22015
User-Agent: curl/7.36.0
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 02:30:37 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
fdef80b1cbd2d625cdbda39c16fda15f68a3d8c_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
```

```

"data" : [ {
  "instanceID" : 5137,
  "name" : "Execute Remote Command",
  "description" : "Executes a command on the remote execution target server.",
  "tags" : "Windows, Linux, Execute Script",
  "serviceTemplateName" : "Execute Remote Command",
  "createTime" : "2015-08-07T14:44:07.000+09:00",
  "modifyTime" : "2015-08-07T14:44:07.000+09:00",
  "serviceState" : "test",
  "serviceGroupName" : "DefaultServiceGroup",
  "iconURL" : "http://10.196.184.182:22015/Automation/icon/services/
com.hitachi.software.dna.cts/remoteCommandExe/01.20.00",
  "vendorName" : "Hitachi, Ltd.",
  "version" : "01.20.00",
  "favorite" : false,
  "failedCount" : 0,
  "completedCount" : 0,
  "executedCount" : 0,
  "latest" : true,
  "supportedScheduleType" : "immediate, schedule, recurrence",
  "submitCount" : 0,
  "serviceTemplateID" : 5106,
  "serviceGroupID" : 3,
  "supportedActionType" : "forciblyStop, retry"
} ],
"count" : 1
}

```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.5.2 Acquisition of service information

### Function

Acquires information about the specified service.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Services/id
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : instance-id,
  "name" : "service-display-name",
  "description" : "description-text",
  "tags" : "tags"
  "serviceTemplateName" : "service-template-name"
  "createTime" : "created-date-and-time",
  "modifyTime" : "updated-date-and-time",
  "serviceState" : "service-state"
  "serviceGroupName" : "service-group-name",
  "iconURL" : "icon-URL",
  "vendorName" : "vendor-name",
  "version" : "version"
  "lastSubmitTime" : "last-submit-time",
  "favorite" : {true|false},
  "failedCount" : failed-count,
  "completedCount" : completed-count,
  "lastFailedTime" : last-failed-time,
  "resetTime" : reset-time,
  "executedCount" : executed-count,
  "latest" : {true|false},
  "imageURL" : "image-URL",
  "supportedScheduleType" : "supported-schedule-type",
  "submitCount" : submit-count,
  "serviceTemplateID" : service-template-id,
  "serviceGroupID" : service-group-id,
  "supportedActionType" : supported-action-type
}
```

## Usage example

In the following example, the API acquires information about the service whose instanceID is 2015.

Request header:

```
GET /Automation/v1/objects/Services/2015 HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 11:40:06 GMT
```

```
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
1aa95d66e62d885b5583da3620bd166fd3a3_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : 2015,
  "name" : "testService1",
  "description" : "description",
  "tags" : "",
  "serviceTemplateName" : "testService",
  "createTime" : "2014-07-14T01:16:11.000-0700",
  "modifyTime" : "2014-07-14T04:36:30.000-0700",
  "serviceState" : "release",
  "serviceGroupName" : "DefaultServiceGroup",
  "iconURL" : "http://10.196.184.238:22015/Automation/icon/services/
com.hitachi.software/remoteCommandExe/01.20.00",
  "vendorName" : "Hitachi,Ltd.",
  "version" : "01.20.00",
  "lastSubmitTime" : "2014-07-14T01:16:11.000-0700",
  "favorite" : false,
  "failedCount" : 0,
  "completedCount" : 0,
  "executedCount" : 0,
  "latest" : true,
  "supportedScheduleType" : "immediate,schedule,recurrence",
  "submitCount" : 0,
  "serviceTemplateID" : 5106,
  "serviceGroupID" : 3,
  "supportedActionType" : "forciblyStop,retry"
}
```

## 2.5.3 Editing a service

### Function

Edits the specified service.

You cannot use this API function to change the property values of services. If you want to change property values, see the topic [2.9.6 Batch update of property values](#) or [2.9.8 Update of a property value](#).

Users who have the Submit role can update only the `favorite` property. Users who have the Admin, Develop, or Modify role can update all properties.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

## API version

v1

## Request format

```
PUT http://host:port/Automation/version/objects/Services/id
```

The request schema has the same format as the response body for the API function Acquisition of service information. The following table describes the object that can be specified as *Services* (member).

Table 2-49: Object that can be specified as Services (member)

Function	Resource name	Number	Description
Service	Services	1	Services resource that has the specified ID

The following table describes the properties that must be specified for this object.

Resource name	Member name	Number
Services	name	1
	description	
	tags	
	favorite	
	serviceState	
	supportedScheduleType	

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	The argument is invalid, or the specified service name already exists.
401	Unauthorized	The user does not have login permission.
404	Not found	The user does not have permission to acquire services, or the service does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : instance-id,
  "name" : "service-display-name",
  "description" : "description-text",
```



```

"tags" : "tags"
"serviceTemplateName" : "service-template-name"
"createTime" : "created-date-and-time",
"modifyTime" : "updated-date-and-time",
"serviceState" : "service-state"
"serviceGroupName" : "service-group-name",
"iconURL" : "icon-URL",
"vendorName" : "vendor-name",
"version" : "version"
"lastSubmitTime" : "last-submit-time",
"favorite" : {true|false},
"failedCount" : failed-count,
"completedCount" : completed-count,
"executedCount" : executed-count,
"latest" : {true|false},
"imageURL" : "image-URL",
"supportedScheduleType" : "supported-schedule-type",
"submitCount" : submit-count,
"serviceTemplateID" : service-template-id,
"serviceGroupID" : service-group-id,
"supportedActionType" : supported-action-type
}

```

## Usage example

In the following example, the API function edits the service whose instanceID is 2015.

Request header:

```

PUT /Automation/v1/objects/Services/2015 HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzZGVtOmlhbmFnZXI=
User-Agent: useragent1

```

Response header:

```

HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 11:40:10 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
5929972368348e976584903133f5f8ce93ce2aec_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json

```

Response body:

```

{
  "instanceID" : 2015,
  "name" : "testService1",
  "description" : "description",
  "tags" : "",
  "serviceTemplateName" : "testService",
  "createTime" : "2014-07-14T01:16:11.000-0700",
  "modifyTime" : "2014-07-14T04:36:30.000-0700",
  "serviceState" : "release",
  "serviceGroupName" : "DefaultServiceGroup",

```

```

"iconURL" : "http://10.196.184.238:22015/Automation/icon/services/
com.hitachi.software/remoteCommandExe/01.20.00",
"vendorName" : "Hitachi,Ltd.",
"version" : "01.20.00",
"lastSubmitTime" : "2014-07-14T01:16:11.000-0700",
"favorite" : false,
"failedCount" : 0,
"completedCount" : 0,
"executedCount" : 0,
"latest" : true,
"supportedScheduleType" : "immediate,schedule,recurrence",
"submitCount" : 0,
"serviceTemplateID" : 5106,
"serviceGroupID" : 3,
"supportedActionType" : "forciblyStop,retry"
}

```

## 2.5.4 Deletion of a service

### Function

Deletes the specified service.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
DELETE http://host:port/Automation/version/objects/Services/id
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
204	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have permission to delete services.
409	Conflict	There is a task generated from the applicable service.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Usage example

In the following example, the API function deletes the service whose instanceID is 2015.

Request header:

```
DELETE /Automation/v1/objects/Services/2015 HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Content-Type: application/json
Content-Length: 918
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 204 No Content
Date: Fri, 07 Aug 2015 09:48:51 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
82b94e7adbdb8cebcb060b12f8c32ee2660a34b_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Content-Length: 0
Content-Type: application/json
```

## 2.5.5 Acquisition of a list of operations for a service

### Function

Acquires a list of operations that can be executed for the specified service.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Services/id/actions
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.

Status code	Message	Description
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "data" : [ {
    "name" : "update",
    "href" : "http://host:port/Automation/version/objects/Services/id",
    "method" : "PUT",
    "parameters" : []
  }, {
    "name" : "submit",
    "href" : " http://host:port/Automation/version/objects/Services/id/actions/
submit/invoke",
    "method" : "POST",
    "parameters" : []
  }, {
    "name" : "detailhelp",
    "href" : " http://host:port/Automation/version/objects/Services/id/actions/
detailhelp",
    "method" : "GET",
    "parameters" : []
  }, {
    "name" : "delete",
    "href" : "http://host:port/Automation/version/objects/Services/id",
    "method" : "DELETE",
    "parameters" : []
  }, {
    "name" : "reset",
    "href" : " http://host:port/Automation/version/objects/Services/id/actions/reset/
invoke",
    "method" : "POST",
    "parameters" : []
  }, {
    "name" : "release",
    "href" : " http://host:port/Automation/version/objects/Services/id/actions/
release/invoke",
    "method" : "POST",
    "parameters" : []
  }, {
    "name" : "maintenance",
    "href" : " http://host:port/Automation/version/objects/Services/id/actions/
maintenance/invoke",
    "method" : "POST",
    "parameters" : []
  }, {
    "name" : "disable",
    "href" : " http://host:port/Automation/version/objects/Services/id/actions/
disable/invoke",
    "method" : "POST",
    "parameters" : []
  }, {
    "name" : "applyTemplate",
    "href" : " http://host:port/Automation/version/objects/Services/id/actions
applyTemplate/invoke",
    "method" : "POST",
    "parameters" : []
  }
] ,
}
```

```
"count" : 9
}
```

## Usage example

In the following example, the API function acquires a list of operations that can be executed for the service whose instanceID is 2004.

Request header:

```
GET /Automation/v1/objects/Services/2004/actions HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 04:40:59 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
72fe74c462e2a50793542df0c0589289ce3f3_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "name" : "update",
    "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004",
    "method" : "PUT",
    "parameters" : [ ]
  }, {
    "name" : "submit",
    "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/
actions/submit/invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "detailhelp",
    "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/
actions/detailhelp",
    "method" : "GET",
    "parameters" : [ ]
  }, {
    "name" : "delete",
    "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004",
    "method" : "DELETE",
    "parameters" : [ ]
  }, {
    "name" : "reset",
    "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/
actions/reset/invoke",
    "method" : "POST",
    "parameters" : [ ]
  }
]
```

```

    }, {
      "name" : "release",
      "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/
actions/release/invoke",
      "method" : "POST",
      "parameters" : [ ]
    }, {
      "name" : "maintenance",
      "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/
actions/maintenance/invoke",
      "method" : "POST",
      "parameters" : [ ]
    }, {
      "name" : "disable",
      "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/
actions/disable/invoke",
      "method" : "POST",
      "parameters" : [ ]
    }, {
      "name" : "applyTemplate",
      "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/
actions/applyTemplate/invoke",
      "method" : "POST",
      "parameters" : [ ]
    } ],
    "count" : 9
  }

```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.5.6 Acquisition of information necessary for executing a service

### Function

Acquires information necessary for executing the specified service.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Services/id/actions/submit
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The user does not have a permission to acquire the service, or the service does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "submit",
  "href" : "http://host:port/Automation/version/objects/Services/id/actions/submit/
invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

The following table describes the objects that can be output as *parameters* (member).

Table 2-50: Objects that can be output as parameters (member) (Acquisition of information necessary for executing a service)

Function	Resource name	Number	Description
Schedule	Schedule	1	Execution schedule for the service
List of property values	PropertyValue	0 to n	Input property for the service

## Usage example

In the following example, the API function acquires necessary information as a preparation for executing the service whose instanceID is 2015.

Request header:

```
GET /Automation/v1/objects/Services/2015/actions/submit HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 04:40:59 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
75cdef77cf941edbf5b2934f6afe1e8e18fdb8a_V1o8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
```

Transfer-Encoding: chunked  
Content-Type: application/json

Response body:

```
{
  "name" : "submit",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2015/actions/submit/invoke",
  "method" : "POST",
  "parameters" : [ {
    "name" : "Execute Remote Command",
    "submitter" : "",
    "scheduleType" : "immediate",
    "description" : "",
    "scheduledStartTime" : "2015-07-30T14:51:23.342+09:00",
    "recurrenceInterval" : "daily",
    "recurrenceDayOfWeek" : "",
    "recurrenceDayOfMonth" : "",
    "recurrenceLastDayOfMonth" : false,
    "recurrenceStartDate" : "2015-07-30",
    "recurrenceTime" : "00:00:00",
    "serviceID" : 5137
  }, {
    "instanceID" : 5112,
    "type" : "string",
    "keyName" : "common.targetHost",
    "value" : "",
    "readOnly" : false,
    "hidden" : false,
    "serviceID" : 5137
  }, {
    "instanceID" : 5135,
    "type" : "string",
    "keyName" : "common.remoteCommand",
    "value" : "",
    "readOnly" : false,
    "hidden" : false,
    "serviceID" : 5137
  }, {
    "instanceID" : 5128,
    "type" : "string",
    "keyName" : "common.remoteCommandParameter",
    "value" : "",
    "readOnly" : false,
    "hidden" : false,
    "serviceID" : 5137
  } ]
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
  - [2.5.7 Execution of a service](#)
-



## 2.5.7 Execution of a service

### Function

Executes the specified service.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/objects/Services/id/actions/submit/invoke
```

The following shows the structure of the request body.

```
{
  "name" : "submit",
  "href" : "http://host:port/Automation/version/objects/Services/id/actions/submit/invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

The following table describes the objects that can be specified as *parameters* (member) in the schema of a request.

Table 2-51: Objects that can be specified as parameters (member)

Function	Resource name	Number	Description
Schedule	Schedule	1	Execution schedule of the service
List of property values	PropertyValue	0 to n	Input property for the service

The tables below describe properties that must be specified for these objects. The following members can be specified for properties regardless of when the service is executed (immediate, schedule, or recurrence).

Resource name	Member name	Number
Schedule	name	1
	description	
	scheduleType	
PropertyValue	keyName	0 to n
	value	

If the timing of service execution is Now or Recurring, the following members can be specified for the property.

Resource name	Member name	Number	Whether the property can be specified
Schedule	scheduledStartTime	1	Can be specified when Later is set.

Resource name	Member name	Number	Whether the property can be specified
Schedule	recurrenceInterval	1	Can be specified when Recurring is set.
	recurrenceMinutes		
	recurrenceDayOfWeek		
	recurrenceDayOfMonth		
	recurrenceLastDayOfMonth		
	recurrenceStartDate		
	recurrenceTime		

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
201	Created	Processing has been successfully completed.
400	Bad Request	The argument is invalid.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have a permission for executing the service.
404	Not found	The user does not have a permission for acquiring the service, or the service does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceId" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ]
}
```

The following table describes the objects that can be output as *affectedResources* (member).

Table 2-52: Objects that can be output as affectedResources (member) (Execution of a service)

Output	Resource name	Number	Description
Link to the created schedule	String	1	Link to the created resource for schedule functionality (Schedules)
Link to the created task	String		Link to the created resource for task functionality (Tasks)

## Usage example

In the following example, the API function executes the service whose instanceID is 2015.

Request header:

```
POST /Automation/v1/objects/Services/2015/actions/submit/invoke HTTP/1.1
Host: 10.196.184.182:22015
Accept: application/json
Content-Type: application/json
Content-Length: 811
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 11:45:34 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
72fe74c462e2a50793542df0c0589289ce3f3_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "3d9069ca-444f-4757-b0c5-a57ddd7d44cf",
  "created" : "2014-07-14T04:45:35.293-0700",
  "updated" : "2014-07-14T04:45:35.293-0700",
  "completed" : "2014-07-14T04:45:35.293-0700",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.182:22015/Automation/v1/objects/
Schedules/2025", "http://10.196.184.182:22015/Automation/v1/objects/Tasks/2026" ]
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.5.8 Acquisition of information necessary for resetting the counter for a service

### Function

Acquires information necessary for resetting the counter for the specified service (initialization of statistics).

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Services/id/actions/reset/
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The user does not have permission to acquire services, or the service does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "reset",
  "href" : " http://host:port/Automation/version/objects/Services/id/actions/reset/
invoke",
  "method" : "POST",
  "parameters" : []
}
```

## Usage example

In the following example, the API function acquires information necessary for resetting the counter for the service whose instanceID is 2004.

Request header:

```
GET /Automation/v1/objects/Services/2004/actions/reset HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 04:44:34 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
afc7e78858ad7ff3a8e53c84ac519a7e663b97b4_V1o8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
```

```
Content-Type: application/json
```

Response body:

```
{
  "name" : "reset",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/actions/
reset/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

## 2.5.9 Reset of the counter for a service

### Function

Resets the counter for the specified service (initialization of statistics).

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/objects/Services/id/actions/reset/invoke
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have permission to reset counters.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "reset",
  "href" : " http://host:port/Automation/version/objects/Services/id/actions/reset/
invoke",
}
```

```
"method" : "POST",
"parameters" : []
}
```

## Usage example

In the following example, the API function resets the counter for the service whose instanceID is 2004.

Request header:

```
GET /Automation/v1/objects/Services/2004/actions/reset HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 04:44:34 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
afc7e78858ad7ff3a8e53c84ac519a7e663b97b4_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "reset",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/actions/
reset/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

## 2.5.10 Acquisition of information necessary for the operation to change the status of a service to release

### Function

Acquires information necessary for the operation to change the status of the specified service to release.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Services/id/actions/release
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The user does not have permission to acquire services, or the service does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "release",
  "href" : " http://host:port/Automation/version/objects/Services/id/actions/release/
invoke",
  "method" : "POST",
  "parameters" : []
}
```

## Usage example

In the following example, the API function acquires information necessary for changing the status of the service whose instanceID is 2004 to release.

Request header:

```
GET /Automation/v1/objects/Services/2004/actions/release HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 04:53:56 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
da3af9677bd825b8186bb9d6f0a67f4dbc78d7_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
```

```
Content-Type: application/json
```

Response body:

```
{
  "name" : "release",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/actions/
release/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

## 2.5.11 Change of the status of a service to release

### Function

Changes the status of the specified service to `release`.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Services/id/actions/release
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The service is in a status that cannot be changed to <code>release</code> .
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : "instance-id",
  "created" : "created-date-and-time",
}
```



```
"updated" : "updated-date-and-time",
"completed" : "completed-date-and-time",
"state" : "state",
"affectedResources" : [ {...} ],
"result" : [ {...} ],
"resultType" : "result-type"
}
```

## Usage example

In the following example, the API function changes the status of the service whose instanceID is 2004 to release.

Request header:

```
POST /Automation/v1/objects/Services/2004/actions/release/invoke HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
Content-Type: application/json
Content-Length: 175
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 04:55:39 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
2a40239379d63c60ba2537f856c1673efd23746b_V1o8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "4c63e655-1ec2-4c70-912f-c1d80be59066",
  "created" : "2015-07-30T13:55:39.457+09:00",
  "updated" : "2015-07-30T13:55:39.457+09:00",
  "completed" : "2015-07-30T13:55:39.457+09:00",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.182:22015/Automation/v1/objects/Services/2004" ],
  "result" : [ ]
}
```

## 2.5.12 Acquisition of information necessary for the operation to change the status of a service to maintenance

### Function

Acquires information necessary for the operation to change the status of the specified service to maintenance.

## Execution permissions

Admin role, Develop role, Modify role

## API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Services/id/actions/maintenance
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The user does not have permission to acquire services, or the service does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "maintenance",
  "href" : " http://host:port/Automation/version/objects/Services/id/actions/
maintenance/invoke",
  "method" : "POST",
  "parameters" : []
}
```

## Usage example

In the following example, the API function acquires information necessary for changing the status of the service whose instanceID is 2004 to maintenance.

Request header:

```
GET /Automation/v1/objects/Services/2004/actions/maintenance HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 05:02:47 GMT
```

```

Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
2370bb888129f799683dc8289b0484da547fceb_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json

```

Response body:

```

{
  "name" : "maintenance",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/actions/
maintenance/invoke",
  "method" : "POST",
  "parameters" : [ ]
}

```

## 2.5.13 Change of the status of a service to maintenance

### Function

Changes the status of the specified service to maintenance.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```

POST http://host:port/Automation/version/objects/Services/id/actions/maintenance/
invoke

```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The service is in a status that cannot be changed to maintenance.
412	Precondition failed	The server is not available.

Status code	Message	Description
500	Server-side error	The status of the service cannot be changed, or a server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ],
  "result" : [ {...} ],
  "resultType" : "result-type"
}
```

## Usage example

In the following example, the API function changes the status of the service whose instanceID is 2004 to maintenance.

Request header:

```
POST /Automation/v1/objects/Services/2004/actions/maintenance/invoke HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
Content-Type: application/json
Content-Length: 183
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 05:04:40 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
23916dfb9e33860332c7e7995f78c2f2507dbf_V1o8Y30JdDBUB31jJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "36a53982-ac92-45aa-acea-21ede67b7df2",
  "created" : "2015-07-30T14:04:41.028+09:00",
  "updated" : "2015-07-30T14:04:41.028+09:00",
  "completed" : "2015-07-30T14:04:41.028+09:00",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.182:22015/Automation/v1/objects/Services/2004" ],
```

```
"result" : [ ]
}
```

## 2.5.14 Acquisition of information necessary for the operation to change the status of a service to disabled

### Function

Acquires information necessary for the operation to change the status of the specified service to `disabled`.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Services/id/actions/disable
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "disable",
  "href" : " http://host:port/Automation/version/objects/Services/id/actions/disable/
invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

### Usage example

In the following example, the API function acquires information necessary for changing the status of the service whose `instanceID` is 2004 to `disabled`.

Request header:

```
GET /Automation/v1/objects/Services/2004/actions/disable HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 05:05:53 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
4ceed74c19dfb6a6c289e561e1c23f5a9088f58_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "disable",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2004/actions/
disable/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

## 2.5.15 Change of the status of a service to disabled

### Function

Changes the status of the specified service to disabled.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/objects/Services/id/actions/disable/invoke
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The service is in a status that cannot be changed to disabled.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ],
  "result" : [ {...} ],
  "resultType" : "result-type"
}
```

## Usage example

In the following example, the API function changes the status of the service whose instanceID is 2004 to disabled.

Request header:

```
POST /Automation/v1/objects/Services/2004/actions/disable/invoke HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
Content-Type: application/json
Content-Length: 175
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 05:07:57 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
3bae2e194f9f7417a578e3d18492e9ccf94388_Vl08Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```

{
  "instanceID" : "ff785246-c3c9-425c-87a5-109336e8b387",
  "created" : "2015-07-30T14:07:58.053+09:00",
  "updated" : "2015-07-30T14:07:58.053+09:00",
  "completed" : "2015-07-30T14:07:58.053+09:00",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.182:22015/Automation/v1/objects/Services/2004" ],
  "result" : [ ]
}

```

## 2.5.16 Acquisition of the URL for the details of a service

### Function

Acquires the URL for displaying the details of the specified service.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Services/id/actions/detailhelp
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```

{
  "name" : "export",
  "href" : "Link-to-the-detail-help",
  "method" : "POST",
  "parameters" : []
}

```



## Usage example

In the following example, the API function acquires the URL for displaying the details of the service whose instanceID is 2004.

Request header:

```
GET /Automation/v1/objects/Services/2004/actions/detailhelp HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 05:08:56 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
8e609f55fd6858f17ddc4527cd6f890b79153e2_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "detailhelp",
  "href" : "http://10.196.184.182:22015/Automation/services/custom/000000000000560/remoteCommandExe.html",
  "method" : "GET",
  "parameters" : [ ]
}
```

## 2.5.17 Acquisition of information necessary for changing the version of the service template used by a service

### Function

Acquires information necessary for the operation to change the version of the service template used by the specified service.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Services/id/actions/applyTemplate
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	
403	Forbidden	The user does not have permission to acquire service templates.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "applyTemplate",
  "href" : " http://host:port/Automation/version/objects/Services/id/actions/
applyTemplate/invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

## Usage example

In the following example, the API function acquires information necessary for changing the version of the service template used by the service whose instanceID is 2188.

Request header:

```
GET /Automation/v1/objects/Services/2188/actions/applyTemplate HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 05:21:04 GMT
Server: Cosminexus HTTP Server is not blacklisted
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
95fa1a17b658d5f34912ec64299aadb522e0d6f5_V1o8Y30JdDBUB31jJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
```

```
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "applyTemplate",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/Services/2188/actions/
applyTemplate/invoke",
  "method" : "POST",
  "parameters" : [ {
    "instanceID" : 2111,
    "keyName" : "SP_GenericApplication",
    "displayName" : "Allocate Volumes for Generic Application",
    "iconURL" : "http://10.196.184.182:22015/Automation/icon/services/
com.hitachi.software.dna.cts/SP_GenericApplication/01.14.00",
    "vendorID" : "com.hitachi.software.dna.cts",
    "version" : "01.14.00",
    "vendorName" : "Hitachi, Ltd.",
    "tags" : "Add New Storage",
    "createTime" : "2015-07-30T14:14:29.000+09:00",
    "modifyTime" : "2015-07-30T14:14:29.000+09:00",
    "description" : "Intelligent allocation service that uses sets of volumes from
the associated infrastructure group to be consumed by server(s) running a generic
application",
    "releaseState" : "release",
    "latest" : false,
    "imageUrl" : "http://10.196.184.182:22015/Automation/services/custom/
000000000002111/SP_GenericApplication_overview.png",
    "supportedScheduleType" : "immediate,schedule",
    "needVUP" : false,
    "componentOutdated" : true,
    "usedServices" : 1,
    "usedTemplates" : 0
  } ]
}
```

## 2.5.18 Change of the version of the service template used by a service

### Function

Applies the service template of any version to the specified service.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/objects/Services/id/applyTemplate/invoke
```

The following shows the structure of the request body.

```
{
  "name" : "applyTemplate",
  "href" : "http://host:port/Automation/version/objects/Services/id/applyTemplate/
  invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

The following table describes the object that can be specified as *parameters* (member) in the schema of a request.

Table 2-53: Object that can be specified as parameters (member)

Function	Resource name	Number	Description
Service template	ServiceTemplate	1	Service template to be upgraded

The following table describes the property that must be specified for this object.

Resource name	Member name	Number
ServiceTemplate	insatnceID	1

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the target service template is invalid.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ],
  "result" : [ {...} ],
  "resultType" : "result-type"
}
```

## Usage example

In the following example, the API function changes the version of the service template used by the service whose instanceID is 2188.

Request header:

```
POST /Automation/v1/objects/Services/2188/actions/applyTemplate/invoke HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
Content-Type: application/json
Content-Length: 1199
Expect: 100-continue
```

Response header:

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 05:23:38 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
456eb72dda7029ba9cbdf3dd57233a25247d2717_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "10920fed-ad4f-4be1-9015-bd2066e5312c",
  "created" : "2015-07-30T14:23:38.683+09:00",
  "updated" : "2015-07-30T14:23:38.683+09:00",
  "completed" : "2015-07-30T14:23:38.683+09:00",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.182:22015/Automation/v1/objects/Services/2188" ],
  "result" : [ ]
}
```

## 2.6 Schedule-related APIs

This section describes operations for managing schedule functionality set for tasks.

### 2.6.1 Acquisition of a list of schedules

#### Function

Acquires a list of schedules set for the specified task.

#### Execution permissions

Admin role, Develop role, Modify role, Submit role

#### API version

v1

#### Request format

```
GET http://host:port/Automation/version/objects/Schedules
```

This API acquires a list of all schedules for which the user who executed the API has permissions. By specifying query parameters, you can filter the schedules for which you want to acquire the list. Specify query parameters in the following format:

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-54: List of query parameters that can be specified for the API function Acquisition of a list of schedules

Query parameter	Filter condition
serviceID	Equal to the specified value
serviceGroupID	
serviceTemplateID	
scheduleStatus <sup>#</sup>	Schedule information for unexecuted tasks

#

If you want to acquire schedule information about tasks that have not been executed yet, specify `running` for `scheduleStatus`.

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

#### Example

The following example specifies 2015 for `serviceID` as a query parameter.

```
?serviceID=2015
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

Data that matches the conditions specified by query parameters in a request is returned in the response body. The following shows the schema of the response body.

```
{
  "data" : [ {"member-of-a-resource-for-schedule-functionality(Schedules)" :
value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API acquires a list of all schedules.

Request header:

```
GET /Automation/v1/objects/Schedules HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:25:42 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
1aa95d66e62d885b5583da3620bd166fd3a3_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "instanceID" : 2060,
    "name" : "testService1_20140714044426_Resubmit",
```

```

    "submitter" : "System",
    "scheduleType" : "immediate",
    "createTime" : "2014-07-14T05:19:39.000-0700",
    "modifyTime" : "2014-07-14T05:19:39.000-0700",
    "description" : "",
    "serviceState" : "release",
    "serviceID" : 2015
  }, {
    "instanceID" : 2029,
    "name" : "testService1_20140714045613",
    "submitter" : "System",
    "scheduleType" : "immediate",
    "createTime" : "2014-07-14T04:56:15.000-0700",
    "modifyTime" : "2014-07-14T04:56:15.000-0700",
    "description" : "",
    "serviceState" : "release",
    "serviceID" : 2015
  }, {
    "instanceID" : 2025,
    "name" : "testService1_20140714044426",
    "submitter" : "System",
    "scheduleType" : "immediate",
    "createTime" : "2014-07-14T04:45:34.000-0700",
    "modifyTime" : "2014-07-14T04:45:34.000-0700",
    "description" : "",
    "serviceState" : "release",
    "serviceID" : 2015
  }, {
    "instanceID" : 2056,
    "name" : "Execute remote command_20140714045708",
    "submitter" : "System",
    "scheduleType" : "immediate",
    "createTime" : "2014-07-14T04:57:09.000-0700",
    "modifyTime" : "2014-07-14T04:57:09.000-0700",
    "description" : "",
    "serviceState" : "test",
    "serviceID" : 2040
  }, {
    "instanceID" : 2134,
    "name" : "stop_20140714052330",
    "submitter" : "System",
    "scheduleType" : "immediate",
    "createTime" : "2014-07-14T05:23:32.000-0700",
    "modifyTime" : "2014-07-14T05:23:32.000-0700",
    "description" : "",
    "serviceState" : "test",
    "serviceID" : 2092
  } ],
  "count" : 5
}

```

---

## Related topics

- [2.2.14 Members of resources](#)
-



## 2.6.2 Acquisition of schedule information

### Function

Acquires information about the specified schedule.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Schedules/id
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : instance-id,
  "name" : "registered-service-name",
  "submitter" : "submit-user-name",
  "status" : "status-of-schedule",
  "scheduleType" : "type-of-schedule",
  "createTime" : "created-date-and-time",
  "modifyTime" : "updated-date-and-time",
  "description" : "description-text",
  "scheduledStartTime" : "scheduled-start-time",
  "recurrenceInterval" : "interval-type",
  "recurrenceDayOfWeek" : "interval-of-weekly-job",
  "recurrenceDayOfMonth" : "interval-of-monthly-job",
  "recurrenceLastDayOfMonth" : {true|false},
  "recurrenceStartDate" : "recurrence-start-date",
  "recurrenceTime" : "exec-time-of-day",
  "serviceState" : "service-state",
  "serviceID" : service-id
}
```

## Usage example

In the following example, the API acquires information about the schedule whose instanceID is 2060.

Request header:

```
GET /Automation/v1/objects/Schedules/2060 HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:26:19 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
55fb30b1218f2ceec1b52d59d1b77b267895821_Vl08Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : 2060,
  "name" : "testService1_20140714044426_Resubmit",
  "submitter" : "System",
  "scheduleType" : "immediate",
  "createTime" : "2014-07-14T05:19:39.000-0700",
  "modifyTime" : "2014-07-14T05:19:39.000-0700",
  "description" : "",
  "serviceState" : "release",
  "serviceID" : 2015
}
```

## 2.6.3 Acquisition of a list of operations for a schedule

### Function

Acquires a list of operations that can be executed for the specified schedule.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Schedules/id/actions
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "data" : [ {
    "name" : "cancel",
    "href" : "http://host:/Automation/version/objects/Schedules/id/actions/cancel/
invoke",
    "method" : "POST",
    "parameters" : []
  }, {
    "name" : "suspend",
    "href" : " http://host:port/Automation/version/objects/Schedules/id/actions/
suspend/invoke",
    "method" : "POST",
    "parameters" : []
  }, {
    "name" : "resume",
    "href" : " http://host:port/Automation/version/objects/Schedules/id/actions/
resume/invoke",
    "method" : "POST",
    "parameters" : []
  } ],
  "count" : 3
}
```

## Usage example

In the following example, the API acquires a list of operations that can be executed for the schedule whose instanceID is 2193.

Request header:

```
GET /Automation/v1/objects/Schedules/2193/actions HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOm1hbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:29:28 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
f214b39fba479af17375flad0e052124041ea60_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "name" : "cancel",
    "href" : "http://10.196.184.238:22015/Automation/v1/objects/Schedules/2193/
actions/cancel/invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "suspend",
    "href" : "http://10.196.184.238:22015/Automation/v1/objects/Schedules/2193/
actions/suspend/invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "resume",
    "href" : "http://10.196.184.238:22015/Automation/v1/objects/Schedules/2193/
actions/resume/invoke",
    "method" : "POST",
    "parameters" : [ ]
  } ],
  "count" : 3
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.6.4 Acquisition of information necessary for canceling a schedule

### Function

Acquires information necessary for canceling the specified schedule.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Schedules/id/actions/cancel
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "cancel",
  "href" : "http://host:port/Automation/version/objects/Schedules/id/actions/cancel/
invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

## Usage example

In the following example, the API acquires information necessary for canceling the schedule whose instanceID is 2193.

Request header:

```
GET /Automation/v1/objects/Schedules/2193/actions/cancel HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:34:39 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
f214b39fba479af17375f1ad0e052124041ea60_V1o8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "cancel",
  "href" : "http://10.196.184.238:22015/Automation/v1/objects/Schedules/2193/actions/cancel/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

---

## Related topics

- [2.6.5 Cancellation of a schedule](#)
- 

## 2.6.5 Cancellation of a schedule

### Function

Cancels the specified schedule.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/objects/Schedules/id/actions/cancel/invoke
```

The following shows the structure of the request body.

```
{
  "name" : "cancel",
  "href" : "http://host:port/Automation/version/objects/Schedules/id/actions/cancel/invoke",
  "method" : "POST",
  "parameters" : null
}
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	Created	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.

Status code	Message	Description
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the task is neither Waiting nor Holding.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceId" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ]
}
```

The following table describes the object that can be output as *affectedResources* (member).

Table 2-55: Object that can be output as affectedResources (member) (Cancellation of a schedule)

Output	Resource name	Number	Description
Link to the affected schedule	String	1	Link to the affected resource for schedule functionality (Schedules)

## Usage example

In the following example, the API cancels the schedule whose instanceID is 2193.

Request header:

```
POST /Automation/v1/objects/Schedules/2193/actions/cancel/invoke HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Content-Type: application/json
Content-Length: 172
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:35:22 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
f214b39fba479af17375f1ad0e052124041ea60_V1o8Y30JdDBUB31jJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "7a2924f8-1d5b-4f94-aef0-babccb2eb525",
  "created" : "2014-07-14T05:35:23.113-0700",
  "updated" : "2014-07-14T05:35:23.113-0700",
  "completed" : "2014-07-14T05:35:23.113-0700",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.238:22015/Automation/v1/objects/Schedules/2193" ]
}
```

## 2.6.6 Acquisition of information necessary for pausing a schedule

### Function

Acquires information necessary for pausing the specified schedule.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Schedules/id/actions/suspend
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "suspend",
  "href" : "http://host:port/Automation/version/objects/Schedules/id/actions/suspend/invoke",
  "method" : "POST",
}
```



```
"parameters" : null
}
```

## Usage example

In the following example, the API acquires information necessary for pausing the schedule whose instanceID is 2193.

Request header:

```
GET /Automation/v1/objects/Schedules/2193/actions/suspend HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzZGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:31:38 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
f214b39fba479af17375f1ad0e052124041ea60_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "suspend",
  "href" : "http://10.196.184.238:22015/Automation/v1/objects/Schedules/2193/actions/
suspend/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

---

## Related topics

- [2.6.7 Pause of a schedule](#)
- 

## 2.6.7 Pause of a schedule

### Function

Pauses the specified schedule.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

## API version

v1

## Request format

```
POST http://host:port/Automation/version/objects/Schedules/id/actions/suspend/invoke
```

The following shows the structure of the request body.

```
{
  "name" : "suspend",
  "href" : "http://host:port/Automation/version/objects/Schedules/id/actions/suspend/
invoke",
  "method" : "POST",
  "parameters" : null
}
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	Created	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the task is not Waiting.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceId" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ]
}
```

The following table describes the object that can be output as *affectedResources* (member).

Table 2-56: Object that can be output as affectedResources (member) (Pause of a schedule)

Output	Resource name	Number	Description
Link to the affected schedule	String	1	Link to the affected Schedules resource

## Usage example

In the following example, the API pauses the schedule whose instanceID is 2193.

Request header:

```
POST /Automation/v1/objects/Schedules/2193/actions/suspend/invoke HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Content-Type: application/json
Content-Length: 174
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:32:16 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
f214b39fba479af17375f1ad0e052124041ea60_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "3a6ac368-e49c-49ec-ac5b-380370800551",
  "created" : "2014-07-14T05:32:16.519-0700",
  "updated" : "2014-07-14T05:32:16.519-0700",
  "completed" : "2014-07-14T05:32:16.519-0700",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.238:22015/Automation/v1/objects/
Schedules/2193" ]
}
```

## 2.6.8 Acquisition of information necessary for resuming a schedule

### Function

Acquires information necessary for resuming the specified schedule.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Schedules/id/actions/resume
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Searching for or deleting a resource has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "resume",
  "href" : "http://host:port/Automation/version/objects/Schedules/id/actions/resume/
invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

## Usage example

In the following example, the API acquires information necessary for resuming the schedule whose instanceID is 2193.

Request header:

```
GET /Automation/v1/objects/Schedules/2193/actions/resume HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:33:15 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
f214b39fba479af17375f1ad0e052124041ea60_V1o8Y30JdDBUB31jJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "resume",
  "href" : "http://10.196.184.238:22015/Automation/v1/objects/Schedules/2193/actions/resume/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

---

## Related topics

- [2.6.9 Resume of a schedule](#)
- 

## 2.6.9 Resume of a schedule

### Function

Resumes the specified schedule.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Schedules/id/actions/resume/invoke
```

The following shows the structure of the request body.

```
{
  "name" : "resume",
  "href" : "http://host:port/Automation/version/objects/Schedules/id/actions/resume/invoke",
  "method" : "POST",
  "parameters" : null
}
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	Created	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.

Status code	Message	Description
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the task is neither Waiting nor Holding.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceId" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ]
}
```

The following table describes the object that can be output as *affectedResources* (member).

Table 2-57: Object that can be output as affectedResources (member) (Resume of a schedule)

Output	Resource name	Number	Description
Link to the affected schedule	String	1	Link to the affected resource for schedule functionality (Schedules)

## Usage example

In the following example, the API resumes the schedule whose instanceID is 2193.

Request header:

```
POST /Automation/v1/objects/Schedules/2193/actions/resume/invoke HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Content-Type: application/json
Content-Length: 172
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:33:56 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
f214b39fba479af17375f1ad0e052124041ea60_Vlo8Y30JdDBUB3ljJSVParTjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "a109b95d-e7ef-4982-ab24-2d062b38e088",
  "created" : "2014-07-14T05:33:56.925-0700",
  "updated" : "2014-07-14T05:33:56.925-0700",
  "completed" : "2014-07-14T05:33:56.925-0700",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.238:22015/Automation/v1/objects/
Schedules/2193" ]
}
```

## 2.7 Task-related APIs

This section describes the operations for managing task resources.

### 2.7.1 Acquisition of a list of tasks

#### Function

Acquires a list of tasks.

#### Execution permissions

Admin role, Develop role, Modify role, Submit role

#### API version

v1

#### Request format

```
GET http://host:port/Automation/version/objects/Tasks
```

This API acquires a list of all tasks for which the user who executed the API has permissions. By specifying query parameters, you can filter the tasks for which you want to acquire the list. Specify query parameters in the following format.

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-58: List of query parameters that can be specified for API Acquisition of a list of tasks

Query parameter	Filter condition
serviceID	Equal to the specified value
scheduleID	
serviceGroupID	
serviceTemplateID	
tags	Whether all values are contained. You can specify multiple values by separating them with a comma (,).
q	<p>For the following schema, a full-text search is performed to determine whether the specified value is contained:</p> <ul style="list-style-type: none"><li>• name</li><li>• submitter</li><li>• description</li><li>• serviceName</li><li>• tags</li><li>• notes</li></ul> <p>If you specify multiple values by separating them with a half-width space character, a full-text search is performed to determine whether all of the specified values are contained. This query parameter is not case sensitive.</p>

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).



## Example

The following example specifies 2015 for serviceID as a query parameter.

```
?serviceID=2015
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	The query parameter is invalid.
401	Unauthorized	The user does not have login permission.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

Data that matches the conditions specified by query parameters in a request is returned in the response body. The following shows the schema of the response body.

```
{
  "data" : [ {"member-of-a-resource-for-task-functionality(Tasks)" :
value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API acquires a list of all tasks.

Request header:

```
GET /Automation/v1/objects/Tasks HTTP/1.1
Authorization: Basic c3lzdGVtOm1hbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 31 Jul 2015 02:00:18 GMT
Server: Cosminexus HTTP Server is not blacklisted
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
baa98567d9a18be55be1594ea9677ab1da826a3_V1o8Y30JdDBUB31jJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
```

Content-Type: application/json

Response body:

```
{
  "data" : [ {
    "instanceID" : 3042,
    "name" : "Execute Remote Command_201507311105831",
    "status" : "waiting",
    "scheduledStartTime" : "2015-07-31T11:30:00.000+09:00",
    "submitter" : "System",
    "submitTime" : "2015-07-31T11:00:06.000+09:00",
    "modifyTime" : "2015-07-31T11:00:06.000+09:00",
    "serviceState" : "release",
    "scheduleType" : "schedule",
    "description" : "",
    "serviceName" : "Execute Remote Command",
    "tags" : "Windows,Linux,Execute Script",
    "serviceGroupName" : "DefaultServiceGroup",
    "todo" : false,
    "notes" : "",
    "serviceTemplateID" : 560,
    "scheduleID" : 3020,
    "serviceGroupID" : 3,
    "serviceID" : 2004,
    "supportedActionType" : "forciblyStop,retry"
  } ],
  "count" : 1
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.7.2 Acquisition of task information

### Function

Acquires information about the specified task.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Tasks/id
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : instance-id,
  "name" : "task-name",
  "status" : "task-status",
  "startTime" : "start-date-and-time",
  "completionTime" : "completion-time",
  "scheduledStartTime" : "schedule-start-date-and-time",
  "submitter" : "submit-user-name",
  "submitTime" : "created-date-and-time",
  "modifyTime" : "updated-date-and-time",
  "serviceState" : "service-state",
  "scheduleType" : "schedule-type",
  "description" : "description",
  "serviceName" : "service-name",
  "tags" : "tags",
  "recurrenceInterval" : "recurrenceInterval",
  "recurrenceTime" : "recurrenceTime",
  "recurrenceStartDate" : "recurrenceStartDate",
  "serviceGroupName" : "serviceGroupName",
  "todo" : {true|false},
  "notes" : "notes",
  "stepTime" : "step-time",
  "serviceTemplateID" : service-template-id,
  "scheduleID" : schedule-id,
  "serviceGroupID" : service-group-id,
  "serviceID" : service-id,
  "supportedActionType" : supported-action-type
}
```

## Usage example

In the following example, the API acquires information about the task whose instanceID is 3042.

Request header:

```
GET /Automation/v1/objects/Tasks/3042 HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 31 Jul 2015 02:02:09 GMT
```

```
Server Cosminexus HTTP Server is not blacklisted
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
bb3f961e88fd1fe908176cbea77a395fcd56_V1o8Y30JdDBUB31jJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : 3042,
  "name" : "Execute Remote Command_20150731105831",
  "status" : "waiting",
  "scheduledStartTime" : "2015-07-31T11:30:00.000+09:00",
  "submitter" : "System",
  "submitTime" : "2015-07-31T11:00:06.000+09:00",
  "modifyTime" : "2015-07-31T11:00:06.000+09:00",
  "serviceState" : "release",
  "scheduleType" : "schedule",
  "description" : "",
  "serviceName" : "Execute Remote Command",
  "tags" : "Windows,Linux,Execute Script",
  "serviceGroupName" : "DefaultServiceGroup",
  "todo" : false,
  "notes" : "",
  "serviceTemplateID" : 560,
  "scheduleID" : 3020,
  "serviceGroupID" : 3,
  "serviceID" : 2004,
  "supportedActionType" : "forciblyStop,retry"
}
```

## 2.7.3 Editing a task

### Function

Edits the notes and TODO for the specified task.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
PUT http://host:port/Automation/version/objects/Tasks/id
```

The request schema has the same format as the response body for the API function Acquisition of service information. The following table describes the object that can be specified as *Task* (member).

Table 2-59: Object that can be specified as Task (member)

Function	Resource name	Number	Description
Task	Task	1	Task resource that has the specified ID

The following table describes the properties that must be specified for this object.

Resource name	Member name	Number
Task	notes	1
	todo	

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have permission to edit tasks.
404	Not found	The user does not have permission to acquire tasks, or the task does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : instance-id,
  "name" : "task-name",
  "status" : "task-status",
  "startTime" : "start-date-and-time",
  "completionTime" : "completion-time",
  "scheduledStartTime" : "schedule-start-date-and-time",
  "submitter" : "submit-user-name",
  "submitTime" : "created-date-and-time",
  "modifyTime" : "updated-date-and-time",
  "serviceState" : "service-state",
  "scheduleType" : "schedule-type",
  "description" : "description",
  "serviceName" : "service-name",
  "tags" : "tags",
  "recurrenceInterval" : "recurrenceInterval",
  "recurrenceTime" : "recurrenceTime",
  "recurrenceStartDate" : "recurrenceStartDate",
  "serviceGroupName" : "serviceGroupName",
  "todo" : {true|false},
  "notes" : "notes",
  "stepTime" : "step-time",
  "serviceTemplateID" : service-template-id,
  "scheduleID" : schedule-id,
}
```

```
"serviceGroupID" : service-group-id,  
"serviceID" : service-id,  
"supportedActionType" : supported-action-type  
}
```

## Usage example

In the following example, the API function edits the notes and TODO for the task whose instanceID is 3042.

Request header:

```
PUT /Automation/v1/objects/Tasks/3042 HTTP/1.1  
Authorization: Basic c3lzdGVtOmlhbmFnZXI=  
User-Agent: curl/7.36.0  
Host: 10.196.184.182:22015  
Accept: application/json  
Content-Type: application/json  
Content-Length: 666
```

Response header:

```
HTTP/1.1 200 OK  
Date: Fri, 31 Jul 2015 03:37:03 GMT  
Server: Cosminexus HTTP Server  
Access-Control-Expose-Headers: WWW-Authenticate  
WWW-Authenticate: HSSO  
91351d8e544375a67473e7c7494d1aa7c67b24_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810  
Access-Control-Allow-Origin: *  
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS  
Access-Control-Allow-Credentials: true  
Cache-Control: no-cache  
Transfer-Encoding: chunked  
Content-Type: application/json
```

Response body:

```
{  
  "instanceID" : 3042,  
  "name" : "Execute Remote Command_20150731105831",  
  "status" : "completed",  
  "startTime" : "2015-07-31T11:30:00.000+09:00",  
  "completionTime" : "2015-07-31T11:30:33.000+09:00",  
  "scheduledStartTime" : "2015-07-31T11:30:00.000+09:00",  
  "submitter" : "System",  
  "submitTime" : "2015-07-31T11:00:06.000+09:00",  
  "modifyTime" : "2015-07-31T12:37:03.000+09:00",  
  "serviceState" : "release",  
  "scheduleType" : "schedule",  
  "description" : "",  
  "serviceName" : "Execute Remote Command",  
  "tags" : "Windows,Linux,Execute Script",  
  "serviceGroupName" : "DefaultServiceGroup",  
  "todo" : true,  
  "notes" : "Notes Test",  
  "serviceTemplateID" : 560,  
  "scheduleID" : 3020,  
  "serviceGroupID" : 3,  
  "serviceID" : 2004,  
  "supportedActionType" : "forciblyStop,retry"  
}* Connection #0 to host 10.196.184.182 left intact
```

## 2.7.4 Deletion of a task

### Function

Deletes the specified task. If the specified task is not a debug task, this API function acquires the URL for archiving the task.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
DELETE http://host:port/Automation/version/objects/Tasks/id
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
204	OK	The task was successfully deleted.
303	See Other	The URL for deleting the task was successfully returned as a response. Use the URL shown in the Location response header to archive the task.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have permission to delete tasks.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Usage example

In the following example, the API function deletes the task whose instanceID is 5169.

Request header:

```
DELETE /Automation/v1/objects/Tasks/5169 HTTP/1.1
Authorization: Basic c3lzdGVtOm1hbmFnZXI=
Host: 192.168.146.132:22015
Accept: application/json
User-Agent:useragent1
```

Response header:

```
HTTP/1.1 303 See Other
Date: Fri, 07 Aug 2015 07:38:26 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
```

```
10a7b94b76e0747b63ee8e0828c186a5d95f699_Vlo8Y30JBWoKHUYTEXAMx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Location: http://192.168.146.132:22015/Automation/v1/objects/Tasks/5169/actions/
archive
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: text/html;charset=utf-8

Response body:

<html><head><title>303 See Other</title></head><body><h1>303 See Other</h1></body></
html>
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.7.5 Acquisition of a list of task operations

### Function

Acquires a list of operations that can be executed for the specified task.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Tasks/id/actions
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.



```

{
  "data" : [ {
    "name" : "update",
    "href" : "http://host:port/Automation/version/objects/Tasks/id",
    "method" : "PUT",
    "parameters" : [ ]
  }, {
    "name" : "delete",
    "href" : "http://host:port/Automation/version/objects/Tasks/id",
    "method" : "DELETE",
    "parameters" : [ ]
  }, {
    "name" : "stop",
    "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/stop/
invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "forceStop",
    "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/forceStop/
invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "resubmit",
    "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/resubmit/
invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "archive",
    "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/archive/
invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "response",
    "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/response/
invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "rerunStart",
    "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/
rerunStart/invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "rerunStepStart",
    "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/
rerunStepStart/invoke",
    "method" : "POST",
    "parameters" : [ ]
  } ],
  "count" : 9
}

```

## Usage example

In the following example, the API acquires a list of operations that can be executed for the task whose instanceID is 5169.

Request header:

```
GET /Automation/v1/objects/Tasks/5169/actions HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 07:32:08 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
10fad7e4bd5eb0e56b4740f5efc08e6dc750d972_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "name" : "update",
    "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5169",
    "method" : "PUT",
    "parameters" : [ ]
  }, {
    "name" : "delete",
    "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5169",
    "method" : "DELETE",
    "parameters" : [ ]
  }, {
    "name" : "stop",
    "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5169/actions/stop/invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "forceStop",
    "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5169/actions/forceStop/invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "resubmit",
    "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5169/actions/resubmit/invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "archive",
    "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5169/actions/archive/invoke",
    "method" : "POST",
    "parameters" : [ ]
  }, {
    "name" : "response",
    "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5169/actions/
```

```

response/invoke",
  "method" : "POST",
  "parameters" : [ ]
}, {
  "name" : "rerunStart",
  "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5169/actions/
rerunStart/invoke",
  "method" : "POST",
  "parameters" : [ ]
}, {
  "name" : "rerunStepStart",
  "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5169/actions/
rerunStepStart/invoke",
  "method" : "POST",
  "parameters" : [ ]
} ],
"count" : 9
}

```

## 2.7.6 Acquisition of information necessary for stopping task execution

### Function

Acquires information necessary for stopping execution of the specified task.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Tasks/id/actions/stop
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Searching for or deleting a resource has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "stop",
  "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/stop/
invoke",
  "method" : "POST",
  "parameters" : []
}
```

## Usage example

In the following example, the API acquires information necessary for stopping execution of the task whose instanceID is 2026.

Request header:

```
GET /Automation/v1/objects/Tasks/2026/actions/stop HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:21:37 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
f214b39fba479af17375f1ad0e052124041ea60_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "stop",
  "href" : "http://10.196.184.238:22015/Automation/v1/objects/Tasks/2026/actions/
stop/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

---

## Related topics

- [2.7.7 Stoppage of task execution](#)
- 

## 2.7.7 Stoppage of task execution

### Function

Stops execution of the specified task.

## Execution permissions

Admin role, Develop role, Modify role, Submit role

## API version

v1

## Request format

```
POST http://host:port/Automation/version/objects/Tasks/id/actions/stop/invoke
```

The following shows the structure of the request body.

```
{
  "name" : "stop",
  "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/stop/invoke",
  "method" : "POST",
  "parameters" : null
}
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	Created	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the task is neither In Progress, Waiting for Response, nor Abnormal Detection.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceId" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ]
}
```

The following table describes the object that can be output as *affectedResources* (member).

Table 2-60: Object that can be output as affectedResources (member) (Stoppage of task execution)

Output	Resource name	Number	Description
Link to the affected task	String	1	Link to the updated resource for task functionality (Tasks)

## Usage example

In the following example, the API stops execution of the task whose instanceID is 2026.

Request header:

```
POST /Automation/v1/objects/Tasks/2026/actions/stop/invoke HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Content-Type: application/json
Content-Length: 164
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:23:58 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO d3b775e19041295c9834a332f7936467d94358e_V0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "f550ef02-b4f8-4332-95da-3b685f2cedf8",
  "created" : "2014-07-14T05:23:59.222-0700",
  "updated" : "2014-07-14T05:23:59.222-0700",
  "completed" : "2014-07-14T05:23:59.222-0700",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.238:22015/Automation/v1/objects/Tasks/2026" ]
}
```

## 2.7.8 Acquisition of information necessary for forcibly stopping a task

### Function

Acquires information necessary for forcibly stopping the specified task.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

## API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Tasks/id/actions/forceStop
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "forceStop",
  "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/forceStop/
invoke",
  "method" : "POST",
  "parameters" : []
}
```

## Usage example

In the following example, the API function acquires information necessary for forcibly stopping the task whose instanceID is 5283.

Request header:

```
GET /Automation/v1/objects/Tasks/5283/actions/forceStop HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Authorization: Basic c3lzdGVtOm1hbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 09:57:14 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
9bf53394a45188743ac8b2522efcc67284cd_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
```

```
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "forceStop",
  "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5283/actions/forceStop/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

## 2.7.9 Forced stoppage of a task

### Function

Forcibly stops the specified task.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/objects/Tasks/id/actions/forceStop/invoke
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the task is not In progress, Waiting for response, or Abnormality detected.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.



```

{
  "instanceId" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ],
  "result" : [ ]
}

```

## Usage example

In the following example, the API function forcibly stops the task whose instanceID is 5381.

Request header:

```

POST /Automation/v1/objects/Tasks/5381/actions/forceStop/invoke HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Content-Type: application/json
Content-Length: 175
Authorization: Basic c3lzdGVtOmlhbmFnZXI=

```

```

{
  "name" : "forceStop",
  "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5338/actions/forceStop/invoke",
  "method" : "POST",
  "parameters" : [ ]
}

```

Response header:

```

HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 10:00:39 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
a2e8ab6f7a9c35323fb7d9331735a9419235ebad_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json

```

Response body:

```

{
  "instanceID" : "68451399-53c2-4f6b-bbdd-be025a61ed02",
  "created" : "2015-08-07T19:00:40.025+09:00",
  "updated" : "2015-08-07T19:00:40.025+09:00",
  "completed" : "2015-08-07T19:00:40.025+09:00",
  "state" : "success",
  "affectedResource" : [ "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5381" ],
  "result" : [ ]
}

```

## 2.7.10 Acquisition of information necessary for re-executing a task

### Function

Acquires information necessary for re-executing the specified task.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Tasks/id/actions/resubmit
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Searching for or deleting a resource has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "resubmit",
  "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/resubmit/
  invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

The following table describes the objects that can be output as *parameters* (member).

Table 2-61: Objects that can be output as parameters (member) (Acquisition of information necessary for re-executing a task)

Function	Resource name	Number	Description
Schedule	Schedule	1	Execution schedule of the service
List of property values	PropertyValue	0 to n	Input property of the service

## Usage example

In the following example, the API acquires information necessary for re-executing the task whose instanceID is 2026.

Request header:

```
GET /Automation/v1/objects/Tasks/2026/actions/resubmit HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:03:20 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO c733364e62b52913e477addabfbf8c55f9de831_v0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "resubmit",
  "href" : "http://10.196.184.238:22015/Automation/v1/objects/Tasks/2026/actions/resubmit/invoke",
  "method" : "POST",
  "parameters" : [ {
    "name" : "testService1_20140714044426_Resubmit",
    "submitter" : "System",
    "scheduleType" : "immediate",
    "description" : "",
    "scheduledStartTime" : "2014-07-14T05:03:20.441-07:00",
    "recurrenceInterval" : "daily",
    "recurrenceDayOfWeek" : "",
    "recurrenceDayOfMonth" : "",
    "recurrenceLastDayOfMonth" : false,
    "recurrenceStartDate" : "2014-07-14",
    "recurrenceTime" : "00:00:00",
    "serviceID" : 2015
  }, {
    "instanceID" : 2012,
    "type" : "string",
    "keyName" : "testProp",
    "value" : "defaultValue",
    "serviceID" : 2015
  } ]
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
  - [2.7.11 Re-execution of a task](#)
-

## 2.7.11 Re-execution of a task

### Function

Re-executes the specified task.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/objects/Tasks/id/actions/resubmit/
invoke
```

The following shows the structure of the request body.

```
{
  "name" : "resubmit",
  "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/resubmit/
invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

The following table describes the objects that can be specified as *parameters* (member) in the schema of a request.

Table 2-62: Objects that can be specified as parameters (member)

Function	Resource name	Number	Description
Schedule	Schedule	1	Execution schedule of the service
List of property values	PropertyValue	0 to n	Input property of the service

The following describes the properties that must be specified for these objects.

In the case of common settings:

Table 2-63: In the case of common settings

Resource name	Member name	Number
Schedule	name	1
Schedule	description	
Schedule	scheduleType	
PropertyValue	keyName	0 to n
PropertyValue	value	

In the case of **Now**:

No property needs to be specified.

In the case of **Later**:

Table 2-64: In the case of Later

Resource name	Member name	Number
Schedule	scheduledStartTime	1

In the case of **Recurring**:

Table 2-65: In the case of Recurring

Resource name	Member name	Number
Schedule	recurrenceInterval	1
Schedule	recurrenceMinutes	
Schedule	recurrenceDayOfWeek	
Schedule	recurrenceDayOfMonth	
Schedule	recurrenceLastDayOfMonth	
Schedule	recurrenceStartDate	
Schedule	recurrenceTime	

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	Created	Processing has been successfully completed.
400	Bad Request	The argument is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the task is neither Completed nor Canceled.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ],
  "result" : [ {...} ],
  "resultType" : "result-type"
}
```

The following table describes the objects that can be output as *affectedResources* (member).

Table 2-66: Objects that can be output as affectedResources (member) (Re-execution of a task)

Output	Resource name	Number	Description
Link to the created schedule	String	1	Link to the created resource for schedule functionality (Schedules)
Link to the created task	String		Link to the created resource for task functionality (Tasks)

## Usage example

In the following example, the API re-executes the task whose instanceID is 2026.

Request header:

```
POST /Automation/v1/objects/Tasks/2026/actions/resubmit/invoke HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Content-Type: application/json
Content-Length: 821
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:19:39 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO 5011884058b535482bf6bac7390956be5fc2122_V0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "d2a2284f-9a94-4be0-8813-e5f991762740",
  "created" : "2014-07-14T05:19:40.089-0700",
  "updated" : "2014-07-14T05:19:40.089-0700",
  "completed" : "2014-07-14T05:19:40.089-0700",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.238:22015/Automation/v1/objects/Schedules/2060", "http://10.196.184.238:22015/Automation/v1/objects/Tasks/2063" ],
  "result" : [ ]
}
```

## 2.7.12 Acquisition of information necessary for responding to a task that is in the status Waiting for Response

### Function

Acquires information necessary for responding to a task that is in the status Waiting for Response. Among the steps of the task that has the specified ID, information about the step that was least recently placed in the status Waiting for Response is acquired.

## Execution permissions

Admin role, Develop role, Modify role, Submit role

## API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Tasks/id/actions/response
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the task is not Waiting for response.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "response",
  "href" : " http://host:port/Automation/version/objects/Tasks/id/actions/response/
invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

## Usage example

In the following example, the API function acquires information necessary for responding to the task whose instanceID is 3179.

Request header:

```
GET /Automation/v1/objects/Tasks/3179/actions/response HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 31 Jul 2015 04:36:56 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
13691d353edd835f6f83942ec70f4ae1411a3f_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "response",
  "href" : "http://10.196.184.182:22015/Automation/v1/objects/Tasks/3179/actions/
response/invoke",
  "method" : "POST",
  "parameters" : [ {
    "instanceID" : 3239,
    "dialogText" : "",
    "labelButton0" : "OK",
    "screenURL" : "services/default/index.jsp",
    "taskID" : 3179
  } ]
}
```

## 2.7.13 Response to a task that is in the status Waiting for Response

### Function

Among the steps of the task that has the specified ID, performs a response input for the step that was least recently placed in the status Waiting for Response.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/objects/Tasks/id/actions/response/invoke
```

The following shows the structure of the request body.

```
{
  "name" : "enter Response",
  "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/response/
invoke",
  "method" : "POST",
```



```
"parameters" : [ {...} ]
}
```

The following table describes the object that can be specified as *parameters* (member) in the schema of a request.

Table 2-67: Object that can be specified as parameters (member)

Function	Resource name	Number	Description
Task	ResponseInput	1	Response input

The following table describes the properties that must be specified for this object.

Resource name	Member name	Number
ResponseInput	instanceId	1
	labelbuttonX <sup>#</sup>	
	taskId	

#

X is replaced with a number.

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	The argument is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the task is not Waiting for Response.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ],
  "result" : [ {...} ],
  "resultType" : "result-type"
}
```

## Usage example

In the following example, the API function responds to the task whose instanceID is 3179.

Request header:

```
POST /Automation/v1/objects/Tasks/3179/actions/response/invoke HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
Content-Type: application/json
Content-Length: 329
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 31 Jul 2015 04:42:14 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
5d4cd25fd30d1b8d6b67f2d7b4cc5479a16364f_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "4fb38028-81d7-4573-851a-672e7524a4fc",
  "created" : "2015-07-31T13:42:15.030+09:00",
  "updated" : "2015-07-31T13:42:15.030+09:00",
  "completed" : "2015-07-31T13:42:15.030+09:00",
  "state" : "success",
  "affectedResource" : [ "http://10.196.184.182:22015/Automation/v1/objects/Tasks/
3179" ],
  "result" : [ ]
}
```

## 2.7.14 Acquisition of information necessary for retrying a task (retry from the failed step)

### Function

Specifies a task, and acquires information necessary for retrying the task from the failed step.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Tasks/id/actions/rerunStart
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "rerunStart",
  "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/rerunStart/
invoke",
  "method" : "POST",
  "parameters" : [ {...} ]
}
```

## Usage example

In the following example, the API acquires information necessary for retrying the task whose instanceID is 5381 from the failed step.

Request header:

```
GET /Automation/v1/objects/Tasks/5381/actions/rerunStart HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: useragent1
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 10:16:10 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
f261dfd5d7e3befa74903ab7318a59455a86df3_Vl08Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "rerunStart",
  "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5381/actions/
rerunStart/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

---

## Related topics

- [2.7.15 Retry from the failed step](#)
- 

## 2.7.15 Retry from the failed step

### Function

Specifies a task, and retries the task from the failed step.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/objects/Tasks/id/actions/rerunStart/invoke
```

The following shows the structure of the request body.

```
{
  "name" : "rerunStart",
  "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/rerunStart/
invoke",
  "method" : "POST",
  "parameters" : null
}
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	Created	Processing has been successfully completed.
400	Bad Request	The argument is invalid.
401	Unauthorized	The user does not have login permission.

Status code	Message	Description
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the task is not Failed.
412	Precondition failed	The server is not running.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceId" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ]
  "result" : [ {...} ],
  "resultType" : "result-type"
}
```

The following table describes an object that can be output as *affectedResources* (member).

Table 2-68: Object that can be output as *affectedResources* (member) (Retry from the failed step)

Output	Resource name	Number	Description
Link to the affected task	String	1	Link to the updated resource for task functionality (Tasks)

## Usage example

In the following example, the API retries the task whose instanceID is 5381 from the failed step.

Request header:

```
POST /Automation/v1/objects/Tasks/5381/actions/rerunStart/invoke HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Content-Type: application/json
Content-Length: 177
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 10:19:44 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
dfd342179388629104cd0bb13d288884bed541b_V1o8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
```

```
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "17356cf0-f709-4561-a56b-17a6fbc321e3",
  "created" : "2015-08-07T19:19:44.552+09:00",
  "updated" : "2015-08-07T19:19:44.552+09:00",
  "completed" : "2015-08-07T19:19:44.552+09:00",
  "state" : "success",
  "affectedResource" : [ "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5381" ],
  "result" : [ ]
}
```

## 2.7.16 Acquisition of information necessary for retrying a task (retry from the step after the failed step)

### Function

Specifies a task, and acquires information necessary for retrying the task from the step after the failed step.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/Tasks/id/actions/rerunStepStart
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "rerunStart",
}
```

```
"href" : "http://host:port/Automation/version/objects/Tasks/id/actions/
rerunStepStart/invoke",
"method" : "POST",
"parameters" : [ {...} ]
}
```

## Usage example

In the following example, the API acquires information necessary for retrying the task whose instanceID is 5381 from the step after the failed step.

Request header:

```
GET /Automation/v1/objects/Tasks/5381/actions/rerunStepStart HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Content-Type: application/json
Authorization: Basic c3lzZGVtOm1hbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 10:24:44 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
7abadbb2b4c4d9c1cf18e5465654ef786a9851_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "rerunStepStart",
  "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5381/actions/
rerunStepStart/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

---

## Related topics

- [2.7.17 Retry from the step after the failed step](#)
- 

## 2.7.17 Retry from the step after the failed step

### Function

Specifies a task, and retries the task from the step after the failed step.

## Execution permissions

Admin role, Develop role, Modify role, Submit role

## API version

v1

## Request format

```
POST http://host:port/Automation/version/objects/Tasks/id/actions/rerunStepStart/
invoke
```

The following shows the structure of the request body.

```
{
  "name" : "rerunStepStart",
  "href" : "http://host:port/Automation/version/objects/Tasks/id/actions/
rerunStepStart/invoke",
  "method" : "POST",
  "parameters" : null
}
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	Created	Processing has been successfully completed.
400	Bad Request	The argument is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the task is not Failed.
412	Precondition failed	The server is not running.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceId" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ]
  "result" : [ {...} ],
  "resultType" : "result-type"
}
```



The following table describes the object that can be output as *affectedResources* (member).

Table 2-69: Object that can be output as *affectedResources* (member) (Retry from the step after the failed step)

Output	Resource name	Number	Description
Link to the affected task	String	1	Link to the updated resource for task functionality (Tasks)

## Usage example

In the following example, the API retries the task whose instanceID is 5381 from the step after the failed step.

Request header:

```
POST /Automation/v1/objects/Tasks/5381/actions/rerunStepStart/invoke HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Content-Type: application/json
Content-Length: 185
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 10:29:33 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
c19a775746fbd61d1efd3658d2b4eacadcfe435_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "852af753-989f-4797-a7dc-50faaf07b896",
  "created" : "2015-08-07T19:29:33.562+09:00",
  "updated" : "2015-08-07T19:29:33.562+09:00",
  "completed" : "2015-08-07T19:29:33.562+09:00",
  "state" : "success",
  "affectedResource" : [ "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5381" ],
  "result" : [ ]
}
```

## 2.7.18 Acquisition of information necessary for archiving a task

### Function

Acquires the argument template necessary for archiving the specified task.

## Execution permissions

Admin role, Develop role, Modify role

## API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Tasks/id/actions/archive
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "name" : "archive",
  "href" : " http://host:port/Automation/version/objects/Tasks/id/actions/archive/
invoke",
  "method" : "POST",
  "parameters" : []
}
```

## Usage example

In the following example, the API function acquires information necessary for archiving the task whose instanceID is 5169.

Request header:

```
GET /Automation/v1/objects/Tasks/5169/actions/archive HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 07:28:17 GMT
Server: Cosminexus HTTP Server
```

```
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
58791edf45552caa5592c652b533c730df4b708_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "name" : "archive",
  "href" : "http://192.168.146.132:22015/Automation/v1/objects/Tasks/5169/actions/
archive/invoke",
  "method" : "POST",
  "parameters" : [ ]
}
```

## 2.7.19 Archiving a task

### Function

Archives the specified task.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
POST http://host:port/Automation/version/objects/Tasks/id/actions/archive/invoke
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	The argument is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
409	Conflict	The status of the task is not Completed, Failed, or Canceled.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : "instance-id",
  "created" : "created-date-and-time",
  "updated" : "updated-date-and-time",
  "completed" : "completed-date-and-time",
  "state" : "state",
  "affectedResources" : [ {...} ],
  "result" : [ {...} ],
  "resultType" : "result-type"
}
```

## Usage example

In the following example, the API function archives the task whose instanceID is 5209.

Request header:

```
POST /Automation/v1/objects/Tasks/5209/actions/archive/invoke HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Content-Type: application/json
Content-Length: 171
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 08:15:46 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
aec4a069aea32fe6d59c8325bfae96af27dde14_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : "0fea3bf2-9747-4d29-a4bf-faaddf22076d",
  "created" : "2015-08-07T17:15:46.474+09:00",
  "updated" : "2015-08-07T17:15:46.474+09:00",
  "completed" : "2015-08-07T17:15:46.474+09:00",
  "state" : "success",
  "affectedResource" : [ "http://192.168.146.132:22015/Automation/v1/objects/TaskHistories/5237" ],
  "result" : [ ]
}
```

## 2.7.20 Acquisition of a list of steps

### Function

Among the steps included in the specified task, acquires a list of steps displayed in the **Task Details** window. This API function is for a JP1/AO instance whose version is earlier than V11.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/FlowSteps
```

Make sure that you specify taskID as a query parameter.

By specifying taskID, you can filter the target task. Among the steps included in the task, you can acquire a list of steps displayed in the **Task Details** window. If no query parameter is specified, or if multiple query parameters are specified, an error occurs. Specify a query parameter in the following format:

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-70: List of query parameters that can be specified for the API function Acquisition of a list of steps

Query parameter	Filter condition
taskID	Equal to the specified value

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

### Example

The following example specifies 512 for taskID as a query parameter.

```
?taskID=512
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	Created	Processing has been successfully completed.
400	Bad Request	The argument is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not running.

Status code	Message	Description
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "data" : [ {"member-of-the-resources-for-step-functionality(FlowSteps)" :
value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API acquires a list of steps to be executed by the task whose task ID is 5381.

Request header:

```
GET /Automation/v1/objects/FlowSteps?taskID=5381 HTTP/1.1
Host: 10.196.184.238:22015
User-Agent: curl/7.36.0
Accept: application/json
Authorization: Basic c3lzZGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:51:18 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
flbd56cdd5e340caa0d6f2419205ba81b3317ef_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "instanceID" : "remoteHostCommandExe_2052",
    "name" : "remoteHostCommandExe",
    "startTime" : "2014-07-14 04:57:10",
    "completionTime" : "2014-07-14 04:57:34",
    "jobStatus" : "normal",
    "comment" : "Executes a command on the remote execution target server and
displays the results.",
    "stepStatus" : "complete"
  } ],
  "count" : 1
}
```

## 2.7.21 Acquisition of task logs

### Function

Acquires the logs for the specified task.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/TaskLogs
```

Among all tasks for which the user who executed the API function has permissions, this API function acquires the logs for the task that has the specified taskID. By specifying query parameters, you can filter the tasks for which you want to acquire the list.

This API acquires the logs for the task that has the specified taskID, in the size specified for readSize or smaller, starting from the point specified for the offset. If reverse is specified, the API acquires the logs in the size specified for readSize to the opposite direction from the offset. Specify query parameters in the format below.

Note that logs are acquired line by line. If the log size reaches the specified size, the line being acquired will be discarded.

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-71: List of query parameters that can be specified for API Acquisition of task logs

Query parameter	Filter condition
taskID	Equal to the specified value
readSize	Equal to the specified value (however, if the log size reaches readSize at the middle of a task log, task logs before the task log are acquired.)
offset	Equal to the specified value
reverse <sup>#</sup>	Acquires task logs, starting from the point specified for offset, in the opposite direction.

#

Do not specify any values for reverse.

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

### Example

The following shows an example setting to acquire 1,000-byte task log data for the task whose taskID is 512, starting from 3,000th byte in the opposite direction.

```
?taskID=512&offset=3000&readSize=1000&reverse
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	Created	Processing has been successfully completed.
400	Bad Request	The argument is invalid.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not running.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "data" : [ {"member-of-the-resources-for-task-log-functionality(Tasklogs)" :
value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API function acquires 5,000,000-byte task log data for the task whose taskID is 2052, starting from the beginning (0th byte).

Request header:

```
GET /Automation/v1/objects/TaskLogs?taskID=10042&offset=0&readSize=5000000 HTTP/1.1
Host:192.168.146.132:22015
User-Agent:sample rest client/1.00.0
Accept:application/json
Accept-Language: en
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 26 Oct 2015 02:28:09 GMT
Server: Cosminexus HTTP Server
Cache-Control: no-cache
WWW-Authenticate: HSSO
a36baaf736fd84afdc27aecf1559fcb8620792b_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
```



```

"data" : [ {
  "instanceID" : 10042,
  "text" : "**** Windows Server 2008 R2
6.1                                     TZ=Asia/
Seoul                                2015/10/26 11:22:00.450\r\n
  yyyy/mm/dd hh:mm:ss.sss            pid      tid      message-
id      message (LANG=en)\r\n
7156 2015/10/26 11:22:00.574      Automation      74170687 36EBDFE4 KNAE08001-
I      Started executing plug-in (task name: Execute Remote
Command 20151026112116, task ID: 10042, step ID: /remoteHostCommandExe, execution
ID: @All1).\r\n
7280 2015/10/26 11:22:00.886      Automation      74170687 36EBDFE4 KNAE08129-
I      The general command plug-in started (command: a).\r\n
7282 2015/10/26 11:22:00.886      Automation      74170687 36EBDFE4 KNAE08071-
I      The setting to elevate to root privileges for SSH connections is now
disabled.\r\n
7307 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08131-E
ER     The general command plug-in failed (command: a, plug-in return code: 77).\r\n
7311 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08002-
I      Plug-in execution completed (task name: Execute Remote
Command 20151026112116, task ID: 10042, step ID: /remoteHostCommandExe, execution
ID: @All1, plug-in return code: 77).\r\n
7313 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08004-
I      schema_version=1.1\r\n
7315 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08004-
I      vendor=com.hitachi.software.dna\r\n
7317 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08004-
I      name=ExecuteCommandPlugin\r\n
7319 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08004-
I      version=01.52.01\r\n
7321 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08004-
I      plugin_type=javaClass\r\n
7323 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08004-
I      vendor_display_name=Hitachi, Ltd.\r\n
7325 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08004-
I      display_name=General Command Plug-in\r\n
7327 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08004-
I      short_description=This plug-in executes a command line on the
destination host.\r\n
7329 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08004-
I      tags=Basic,Hitachi\r\n
7331 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08004-
I      enable_SSH_charset_detection=true\r\n
7333 2015/10/26 11:22:10.652      Automation      74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/account, value=\r\n
7335 2015/10/26 11:22:10.668      Automation      74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/commandLine, value=?
dna_common.remoteCommand?\r\n
7337 2015/10/26 11:22:10.668      Automation      74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/commandLineParameter, value=?
dna_common.remoteCommandParameter?\r\n
7339 2015/10/26 11:22:10.668      Automation      74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/credentialType, value=destination\r\n
7341 2015/10/26 11:22:10.668      Automation      74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/destinationHost, value=?
dna_common.targetHost?\r\n
7343 2015/10/26 11:22:10.668      Automation      74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/elevatePrivileges, value=false\r\n
7345 2015/10/26 11:22:10.668      Automation      74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/keyboardInteractiveAuthentication, value=
\r\n
7347 2015/10/26 11:22:10.668      Automation      74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/password, value=*****\r\n
7349 2015/10/26 11:22:10.668      Automation      74170687 36EBDFE4 KNAE08004-

```

```

I      property=/remoteHostCommandExe/publicKeyAuthentication, value=\r\n
7351 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/stdoutPattern1, value=((?s).*)\r\n
7353 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/stdoutPattern2, value=\r\n
7355 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/stdoutPattern3, value=\r\n
7357 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/stdoutProperty1,
value=common.stdoutProperty\r\n
7359 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/stdoutProperty2, value=\r\n
7361 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/stdoutProperty3, value=\r\n
7363 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=/remoteHostCommandExe/suPassword, value=*****\r\n
7365 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.jp1.password, value=*****\r\n
7367 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.jp1.username, value=jp1admin\r\n
7369 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.mail.bcc, value=\r\n
7371 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.mail.cc, value=\r\n
7373 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.mail.from, value=*****\r\n
7375 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.mail.notify, value=false\r\n
7377 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.mail.smtp.password, value=*****
\r\n
7379 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.mail.smtp.port, value=25\r\n
7381 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.mail.smtp.server, value=\r\n
7383 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.mail.smtp.userid, value=\r\n
7385 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.mail.to, value=\r\n
7387 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.ssh.privatekey.passphrase,
value=*****\r\n
7389 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=com.hitachi.software.dna.sys.task.log.level, value=10\r\n
7391 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=common.remoteCommand, value=a\r\n
7393 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=common.remoteCommandParameter, value=\r\n
7395 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=common.stdoutProperty, value=\r\n
7397 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=common.targetHost, value=a\r\n
7399 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=foreach.max_value, value=3\r\n
7401 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=mail.plugin.retry.interval, value=10\r\n
7403 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=mail.plugin.retry.times, value=3\r\n
7405 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=plugin.private.builtin.baseUrl, value=http://WIN-FC6MCPD47CQ:
22015/Automation/\r\n
7407 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I      property=reserved.service.category, value=Execute Script, Linux, Windows\r
\n

```

```

7409 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.service.name, value=Execute Remote Command\r\n
7411 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.service.resourceGroupName, value=Default Service Group
\r\n
7413 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.service.serviceGroupName, value=Default Service Group\r
\n
7415 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.step.path, value=/remoteHostCommandExe\r\n
7417 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.step.prevReturnCode, value=0\r\n
7419 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.task.description, value=\r\n
7421 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.task.dir, value=C:\\Program Files\\Hitachi\\JPlAO\\data
\\task\\10042\r\n
7423 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.task.id, value=10042\r\n
7425 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.task.name, value=Execute Remote
Command_20151026112116\r\n
7427 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.task.submitter, value=System\r\n
7429 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.task.tags, value=Execute Script,Linux,Windows\r\n
7431 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=reserved.task.url, value=http://WIN-FC6MCPD47CQ:22015/
Automation/launcher/TaskDetails?task_id=10042\r\n
7433 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=account, value=\r\n
7435 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=commandLine, value=a\r\n
7437 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=commandLineParameter, value=\r\n
7439 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=credentialType, value=destination\r\n
7441 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=destinationHost, value=a\r\n
7443 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=elevatePrivileges, value=false\r\n
7445 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=keyboardInteractiveAuthentication, value=\r\n
7447 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=password, value=*****\r\n
7449 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=publicKeyAuthentication, value=\r\n
7451 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=stdoutPattern1, value=((?s).*)\r\n
7453 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=stdoutPattern2, value=\r\n
7455 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=stdoutPattern3, value=\r\n
7457 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08004-
I property=suPassword, value=*****\r\n
7459 2015/10/26 11:22:10.668 Automation 74170687 36EBDFE4 KNAE08009-
I No standard output exists.\r\n
7461 2015/10/26 11:22:10.746 Automation 74170687 36EBDFE4 KNAE08014-E
ER Cannot resolve the specified host name (error details: a [errno=11004,
syscall=getaddrinfo]). The specified host name could not be resolved. Check your
network and DNS configuration, and then re-execute the service.\r\n
7463 2015/10/26 11:22:10.746 Automation 74170687 36EBDFE4 KNAE08016-E
ER An error occurred while executing the plug-in (task name: Execute Remote
Command_20151026112116, task ID: 10042, step ID: /remoteHostCommandExe, execution

```

ID: @A111, plug-in return code: 77). The possible causes are as follows\r\n

- (1) An error occurred during plug-in execution.\r\n
- (2) An operation was performed to forcibly stop the task.\r\n
- (3) An operation was performed to stop the product.\r\n

In the dialog box or in Server[n].log, refer to the error message that was output before and after the error occurred, and take the appropriate action. If there is no evidence of an operation to forcibly stop a task or to stop the product, and if no error message was output before or after the error occurred, use the data collection tool to collect the necessary information, and then contact your system administrator.\r\n

```
"  
  "totalSize" : 13065,  
  "readSize" : 13065,  
  "lineCount" : 88,  
  "offset" : 0,  
  "reverse" : false  
} ],  
"count" : 1  
}
```

## 2.8 List of history-related API functions

### 2.8.1 Acquisition of a list of history records

#### Function

Acquires a list of history records.

#### Execution permissions

Admin role, Develop role, Modify role, Submit role

#### API version

v1

#### Request format

```
GET http://host:port/Automation/version/objects/TaskHistories
```

This API function acquires a list of all history records for which the user who executed the API function has permissions. By specifying query parameters, you can filter the history records for which you want to acquire the list. Specify query parameters in the following format:

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-72: List of query parameters that can be specified for the API function Acquisition of a list of history records

Query parameter	Filter condition
start	Whether startTime is equal to or later than the specified value
end	Whether completionTime is equal to or earlier than the specified value
serviceGroupID	Equal to the specified value
tags	Whether all values are contained. You can specify multiple values by separating them with a comma (,).
q	<p>For the following schema, a full-text search is performed to determine whether the specified value is contained:</p> <ul style="list-style-type: none"><li>• name</li><li>• submitter</li><li>• serviceName</li><li>• tags</li><li>• description</li><li>• notes</li></ul> <p>If you specify multiple values by separating them with a half-width space character, a full-text search is performed to determine whether all of the specified values are contained. This query parameter is not case sensitive.</p>

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

Data that matches the conditions specified by query parameters in a request is returned in the response body. The following shows the schema of the response body.

```
{
  "data" : [ {"member-of-the-resources-for-history-functionality(TaskHistories)" :
value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API function acquires a list of all history records.

Request header:

```
GET /Automation/v1/objects/TaskHistories HTTP/1.1
Authorization: Basic c3lzZdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 31 Jul 2015 06:22:25 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
24f87c98d12f4f434cf398edcbe582939cee4d6_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "instanceID" : 4006,
    "name" : "Execute Remote Command_20150731105831",
```

```

"submitter" : "System",
"serviceName" : "Execute Remote Command",
"tags" : "Windows,Linux,Execute Script",
"scheduleType" : "schedule",
"scheduledStartTime" : "2015-07-31T11:30:00.000+09:00",
"startTime" : "2015-07-31T11:30:00.000+09:00",
"completionTime" : "2015-07-31T11:30:33.000+09:00",
"archiveTime" : "2015-07-31T15:22:21.000+09:00",
"taskID" : 3042,
"submitTime" : "2015-07-31T11:00:06.000+09:00",
"status" : "completed",
"description" : "",
"serviceState" : "release",
"todo" : true,
"notes" : "Notes Test",
"serviceGroupName" : "DefaultServiceGroup",
"serviceGroupID" : 3
} ],
"count" : 1
}

```

## Related topics

- [2.2.14 Members of resources](#)

## 2.8.2 Deletion of history records (with conditions specified)

### Function

Deletes history records according to the conditions specified by query parameters.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
DELETE http://host:port/Automation/version/objects/TaskHistories
```

By specifying query parameters, you can filter the history records to be deleted. Specify query parameters in the following format:

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-73: List of query parameters that can be specified for the API function Deletion of history records (with conditions specified)

Query parameter	Filter condition
start	Whether startTime is equal to or later than the specified value

Query parameter	Filter condition
end	Whether completionTime is equal to or earlier than the specified value
serviceGroupID	Equal to the specified value

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
204	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have permission to delete history records.
412	Precondition failed	The server is not running.
500	Server-side error	A server processing error occurred.

## Usage example

In the following example, the API function deletes the history records for which serviceGroupID is 1451 and the period is between July 31 and August 31 in 2015.

Request header:

```
DELETE /Automation/v1/objects/TaskHistories?
serviceGroupID=1451&start=2015-07-31T11:30:00.000+09:00&end=2015-08-31T11:30:00.000+0
9:00 HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 204 No Content
Date: Fri, 07 Aug 2015 11:17:40 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
e949c7e079a0bc9a137cd1bf3515c72685a506a_V1o8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Content-Length: 0
Content-Type: application/json
```

## Related topics

- [2.2.14 Members of resources](#)



## 2.8.3 Acquisition of a history record

### Function

Acquires the history record that has the specified ID.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/TaskHistories/id
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : instance-id,
  "name" : "task-name",
  "submitter" : "submit-user-name",
  "serviceName" : "service-name",
  "tags" : "tags",
  "scheduleType" : "type-of-schedule",
  "scheduledStartTime" : "schedule-start-date-and-time",
  "startTime" : "start-date-and-time",
  "completionTime" : "completion-date-and-time",
  "stepStartTime" : "step-start-time",
  "recurrenceInterval" : "interval-type",
  "recurrenceDayOfWeek" : "interval-of-weekly-job",
  "recurrenceDayOfMonth" : "interval-of-monthly-job",
  "recurrenceLastDayOfMonth" : {true|false},
  "recurrenceTime" : "exec-time-of-day",
  "archiveTime" : "removed-date-and-time",
  "taskID" : task-id,
  "submitTime" : "submit-date-and-time",
  "recurrenceStartDate" : "recurrence-start-date-and-time",
```

```
"status" : "task-status",
"description" : "description",
"serviceState" : "service-state",
"todo" : {true|false},
"notes" : "notes",
"serviceGroupName" : "service-group-name",
"serviceGroupID" : service-group-id
}
```

## Usage example

In the following example, the API function acquires the history record whose instanceID is 4006.

Request header:

```
GET /Automation/v1/objects/TaskHistories/4006 HTTP/1.1
Authorization: Basic c3lzZGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 31 Jul 2015 06:24:06 GMT
Server: Cosminexus HTTP Server is not blacklisted
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
2615a636c3da92888fe355da9ca7d223e6e214_V1o8Y30JdDBUB31jJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : 4006,
  "name" : "Execute Remote Command_201507311105831",
  "submitter" : "System",
  "serviceName" : "Execute Remote Command",
  "tags" : "Windows,Linux,Execute Script",
  "scheduleType" : "schedule",
  "scheduledStartTime" : "2015-07-31T11:30:00.000+09:00",
  "startTime" : "2015-07-31T11:30:00.000+09:00",
  "completionTime" : "2015-07-31T11:30:33.000+09:00",
  "archiveTime" : "2015-07-31T15:22:21.000+09:00",
  "taskID" : 3042,
  "submitTime" : "2015-07-31T11:00:06.000+09:00",
  "status" : "completed",
  "description" : "",
  "serviceState" : "release",
  "todo" : true,
  "notes" : "Notes Test",
  "serviceGroupName" : "DefaultServiceGroup",
  "serviceGroupID" : 3
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.8.4 Deletion of history records (with an ID specified)

### Function

Deletes the history record by specifying a task ID.

### Execution permissions

Admin role, Develop role, Modify role

### API version

v1

### Request format

```
DELETE http://host:port/Automation/version/objects/TaskHistories/id
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
204	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have permission to delete history records.
412	Precondition failed	The server is not running.
500	Server-side error	A server processing error occurred.

### Usage example

In the following example, the API function deletes the history record for the task whose instanceID is 5237.

```
Request header:
```

```
DELETE /Automation/v1/objects/TaskHistories/5237 HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

```
Response header:
```

```
HTTP/1.1 204 No Content
Date: Fri, 07 Aug 2015 11:14:12 GMT
Server: Cosminexus HTTP Server
```

```
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
3b6cddc1eaffe8cd8c2bbcc88ce991e8419472cc_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Content-Length: 0
Content-Type: application/json
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.8.5 Acquisition of a list of operations for a history record

### Function

Acquires a list of operations that can be executed for the history record that has the specified ID.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/TaskHistories/id/actions
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "data" : [ {
```

```
    "name" : "delete",
    "href" : "http://host:port/Automation/version/objects/TaskHistories/id",
    "method" : "DELETE",
    "parameters" : [ ]
  } ],
  "count" : 1
}
```

## Usage example

In the following example, the API function acquires a list of operations that can be performed for the history record whose instanceID is 5237.

Request header:

```
GET /Automation/v1/objects/TaskHistories/5237/actions HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 11:12:20 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
a754baf585ff2447abf34a09fb93ea3b953cfe_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "name" : "delete",
    "href" : "http://192.168.146.132:22015/Automation/v1/objects/TaskHistories/5237",
    "method" : "DELETE",
    "parameters" : [ ]
  } ],
  "count" : 1
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
-

## 2.9 Property-related APIs

---

This section describes the operations for managing property definitions or property values.

### 2.9.1 Acquisition of a list of property definitions

#### Function

Acquires a list of property definitions. The API function targets service properties for which the input/output type is in or out, or service share properties.

#### Execution permissions

Admin role, Develop role, Modify role, Submit role

#### API version

v1

#### Request format

```
GET http://host:port/Automation/version/objects/PropertyDefinitions
```

This API acquires a list of property definitions for all services and tasks for which the user who executed the API has permissions. By specifying query parameters, you can filter the property definitions for which you want to acquire the list. Specify query parameters in the following format:

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-74: List of query parameters that can be specified for API Acquisition of a list of property definitions

Query parameter	Filter condition
serviceID	Equal to the specified value
taskID	
serviceTemplateID	

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

#### Example

The following example specifies 16731 for serviceID and 512 for taskID as query parameters.

```
?serviceID=16731&taskID=512
```

#### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

Data that matches the conditions specified by query parameters in a request is returned in the response body. The following shows the schema of the response body.

```
{
  "data" : [ {"member-of-the-resources-for-property-definition-
functionality(PropertyDefinitions)" : value ... }, ... ],
  "count" : number-of-data-items-that-matches-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API acquires a list of property definitions for all services and tasks.

Request header:

```
GET /Automation/v1/objects/PropertyDefinitions HTTP/1.1
Host:192.168.146.132:22015
User-Agent:sample rest client/1.00.0
Accept:application/json
Accept-Language: en
Authorization: Basic c3lzZGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 26 Oct 2015 02:47:46 GMT
Server: Cosminexus HTTP Server
Cache-Control: no-cache
WWW-Authenticate: HSSO
552c3db4cc540ed80ae43b191bde72ec914673_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "instanceID" : 9002,
    "keyName" : "common.targetHost",
    "displayName" : "Host name of execution target server",
    "defaultValue" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
```

```

    "description" : "Specifies the host name or IP address of the execution target
server. IPv6 addresses are not supported.",
    "mode" : "in",
    "required" : true,
    "maxLength" : 255,
    "minLength" : 1,
    "pattern" : "^[0-9a-zA-Z\\.\-]*$",
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 9033
}, {
    "instanceID" : 9097,
    "keyName" : "common.targetHost",
    "displayName" : "Host name of execution target server",
    "defaultValue" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Specifies the host name or IP address of the execution target
server. IPv6 addresses are not supported.",
    "mode" : "in",
    "required" : true,
    "maxLength" : 255,
    "minLength" : 1,
    "pattern" : "^[0-9a-zA-Z\\.\-]*$",
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 9095
}, {
    "instanceID" : 5513,
    "keyName" : "service.errorMessage",
    "displayName" : "Summary Message",
    "defaultValue" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Stores a summary message of the task execution results.",
    "mode" : "out",
    "required" : false,
    "propertyGroupName" : "System_Properties",
    "validationScript" : "",
    "readOnly" : true,
    "hidden" : true,
    "reference" : false,
    "serviceTemplateID" : 5485
}, {
    "instanceID" : 5715,
    "keyName" : "service.errorMessage",
    "displayName" : "Error message",
    "defaultValue" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Set the error message displayed in the Task Details window in
the Messages area.",
    "mode" : "out",
    "required" : false,
    "propertyGroupName" : "reserved.defaultGroup",

```



```

    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 5658
  }, {
    "instanceID" : 6087,
    "keyName" : "service.errorMessage",
    "displayName" : "Error message",
    "defaultValue" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Set the error message displayed in the Task Details window in
the Messages area.",
    "mode" : "out",
    "required" : false,
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 6096
  }, {
    "instanceID" : 6183,
    "keyName" : "service.errorMessage",
    "displayName" : "Error message",
    "defaultValue" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Set the error message displayed in the Task Details window in
the Messages area.",
    "mode" : "out",
    "required" : false,
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 6178
  }, {
    "instanceID" : 9306,
    "keyName" : "service.errorMessage",
    "displayName" : "service.errorMessage",
    "defaultValue" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "",
    "mode" : "out",
    "required" : false,
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 9303
  }, {
    "instanceID" : 5594,
    "keyName" : "/FP_GenericApplication/service.errorMessage",
    "displayName" : "Summary Message",
    "defaultValue" : "",
    "type" : "string",

```

```

"visibility" : "exec",
"scope" : "local",
"description" : "Stores a summary message of the task execution results.",
"mode" : "out",
"required" : false,
"propertyGroupName" : "/FP_GenericApplication/System_Properties",
"validationScript" : "",
"readOnly" : true,
"hidden" : true,
"reference" : false,
"serviceTemplateID" : 5658
}, {
  "instanceID" : 6177,
  "keyName" : "/localeTest/plugin.destinationHost",
  "displayName" : "display name:Destination host",
  "defaultValue" : "",
  "type" : "string",
  "visibility" : "config",
  "scope" : "local",
  "description" : "For this property, specify the IPv4 address, IPv6 address, or
host name of the target host. You must specify a host that is part of a network
configuration in which the server and the target host are able to communicate
directly.",
  "mode" : "in",
  "required" : true,
  "propertyGroupName" : "reserved.defaultGroup",
  "validationScript" : "",
  "readOnly" : false,
  "hidden" : false,
  "reference" : false,
  "serviceTemplateID" : 6178
}, {
  "instanceID" : 9029,
  "keyName" : "common.remoteCommand",
  "displayName" : "Command",
  "defaultValue" : "",
  "type" : "string",
  "visibility" : "exec",
  "scope" : "local",
  "description" : "Specify the full path of the command to be executed on the
execution target server. If the path contains a space, enclose the entire path in
double quotation marks.",
  "mode" : "in",
  "required" : true,
  "maxLength" : 256,
  "minLength" : 1,
  "propertyGroupName" : "reserved.defaultGroup",
  "validationScript" : "",
  "readOnly" : false,
  "hidden" : false,
  "reference" : false,
  "serviceTemplateID" : 9033
}, {
  "instanceID" : 9136,
  "keyName" : "common.remoteCommand",
  "displayName" : "Command",
  "defaultValue" : "",
  "type" : "string",
  "visibility" : "exec",
  "scope" : "local",
  "description" : "Specify the full path of the command to be executed on the
execution target server. If the path contains a space, enclose the entire path in
double quotation marks.",
  "mode" : "in",

```

```

    "required" : true,
    "maxLength" : 256,
    "minLength" : 1,
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 9095
  }, {
    "instanceID" : 5515,
    "keyName" : "fileProvisioning.nfsSetting.nfsEnable",
    "displayName" : "Enable NFS Provisioning",
    "defaultValue" : "true",
    "type" : "boolean",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Enable NFS",
    "mode" : "in",
    "required" : false,
    "propertyGroupName" : "NFS_Settings",
    "validationScript" : "",
    "readOnly" : true,
    "hidden" : true,
    "reference" : false,
    "serviceTemplateID" : 5485
  }, {
    "instanceID" : 10,
    "keyName" : "com.hitachi.software.dna.sys.mail.notify",
    "displayName" : "Email notification",
    "defaultValue" : "false",
    "type" : "boolean",
    "visibility" : "config",
    "scope" : "share",
    "description" : "Enables or disables the email notification functionality.
(Built-in shared service property)",
    "mode" : "in",
    "required" : true,
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 9024,
    "keyName" : "common.remoteCommandParameter",
    "displayName" : "Command parameters",
    "defaultValue" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Specify the parameters for the command to be executed on the
execution target server. If a parameter contains a space, enclose the entire
parameter in double quotation marks.",
    "mode" : "in",
    "required" : false,
    "maxLength" : 1024,
    "minLength" : 1,
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 9033
  }, {
    "instanceID" : 9137,
    "keyName" : "common.remoteCommandParameter",

```

```

    "displayName" : "Command parameters",
    "defaultValue" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Specify the parameters for the command to be executed on the
execution target server. If a parameter contains a space, enclose the entire
parameter in double quotation marks.",
    "mode" : "in",
    "required" : false,
    "maxLength" : 1024,
    "minLength" : 1,
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 9095
}, {
    "instanceID" : 5690,
    "keyName" : "/FP_GenericApplication/fileProvisioning.nfsSetting.nfsEnable",
    "displayName" : "Enable NFS Provisioning",
    "defaultValue" : "true",
    "type" : "boolean",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Enable NFS",
    "mode" : "in",
    "required" : false,
    "propertyGroupName" : "/FP_GenericApplication/NFS_Settings",
    "validationScript" : "",
    "readOnly" : true,
    "hidden" : true,
    "reference" : false,
    "serviceTemplateID" : 5658
}, {
    "instanceID" : 5402,
    "keyName" : "fileProvisioning.nfsSetting.nfsPathOption",
    "displayName" : "Path Options",
    "defaultValue" : "true",
    "type" : "boolean",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "",
    "mode" : "in",
    "required" : false,
    "propertyGroupName" : "NFS_Settings",
    "validationScript" : "",
    "readOnly" : true,
    "hidden" : true,
    "reference" : false,
    "serviceTemplateID" : 5485
}, {
    "instanceID" : 5,
    "keyName" : "com.hitachi.software.dna.sys.mail.smtp.server",
    "displayName" : "SMTP server address",
    "defaultValue" : "",
    "type" : "string",
    "visibility" : "config",
    "scope" : "share",
    "description" : "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)",

```

```
"mode" : "in",
"required" : false,
"maxLength" : 255,
"minLength" : 0,
"readOnly" : false,
"hidden" : false
} ],
"count" : 18
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.9.2 Acquisition of property definition information

### Function

Acquires information about the specified property definition.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/PropertyDefinitions/id
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The requested resource or operation does not exist, or the user does not have read permission for the resource.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : instance-id,
```

```

"keyName" : "key-name",
"displayName" : "display-name",
"defaultValue" : "default-value",
"type" : "type",
"visibility" : "visibility",
"scope" : "scope",
"description" : "description",
"mode" : "mode",
"required" : {true|false},
"maxLength" : max-length,
"minLength" : min-length,
"minValue" : min-value,
"maxValue" : max-value,
"pattern" : "pattern",
"valueList" : "value-list",
"propertyGroupName" : "property-group-name",
"validationScript" : "validation-script",
"readOnly" : {true|false},
"hidden" : {true|false},
"reference" : {true|false},
"serviceTemplateID" : service-template-id,
}

```

## Usage example

In the following example, the API acquires information about the property definition whose instanceID is 158.

Request header:

```

GET /Automation/v1/objects/PropertyDefinitions/158 HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1

```

Response header:

```

HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:37:24 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO 3096d91c11fd92d841b3513ed988ba758237cd1_V0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json

```

Response body:

```

{
  "instanceID" : 158,
  "keyName" : "remoteHost",
  "displayName" : "Remote Host",
  "defaultValue" : "",
  "description" : "Specify the IP address or host name of the remote host. The
remote host must be in a network environment that can communicate with the server.
You cannot specify more than one remotehost.",
  "mode" : "in",
  "required" : true,
  "maxLength" : 255,

```

```

"minLength" : 1,
"pattern" : "^[0-9a-zA-Z\\.\\"-]*$",
"propertyGroupName" : "reserved.defaultGroup",
"validationScript" : "",
"readOnly" : false,
"hidden" : false,
"reference" : false,
"serviceTemplateID" : 5106
}

```

## 2.9.3 Acquisition of a list of operations for a property definition

### Function

Acquires a list of operations that can be executed for the specified property definition.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/PropertyDefinitions/id/actions
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The request conflicts with another request, or the request does not match the current status of the object.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```

{
  "data" : [ {"member-of-the-resources-for-property-definition-
functionality(PropertyDefinitions)" : value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}

```

## Usage example

In the following example, the API acquires a list of operations that can be executed for the property definition whose instanceID is 158.

Request header:

```
GET /Automation/v1/objects/PropertyDefinitions/158/actions HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:38:20 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO 78d4d9d37740a76bfe7212277228eb2db759bb10_V0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ ],
  "count" : 0
}
```

## 2.9.4 Acquisition of lists of property definitions and property values

### Function

Acquires lists of property definitions and property values.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/PropertyInformations
```

When executing the API function, make sure that you specify query parameters to filter property definitions and property values for which you want to acquire the lists. Specify query parameters in the following format:



```
?query-parameter=value[&query-parameter=value...]
```

Table 2-75: Query parameters that can be specified for the API function Acquisition of lists of property definitions and property values

Query parameter	Filter condition
serviceID	Equal to the specified value
taskID	
scheduleID	
shared	Targets service share properties.

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "data" : [ {"member-of-the-resources-for--property-definition-and-property-value-
functionality(PropertyInformations)" : value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API function acquires the lists of property definitions and property values for the service whose serviceID is 2004.

Request header:

```
GET /Automation/v1/objects/PropertyInformations?serviceID=2004 HTTP/1.1
Authorization: Basic c3lzZGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 31 Jul 2015 06:27:14 GMT
```

```
Server Cosminexus HTTP Server is not blacklisted
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
94728cefd3f4c996534144711565199189dd8_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "instanceID" : 2010,
    "keyName" : "common.targetHost",
    "displayName" : "Host name of execution target server",
    "defaultValue" : "",
    "value" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Specifies the host name or IP address of the execution target
server. IPv6 addresses are not supported.",
    "mode" : "in",
    "required" : true,
    "maxLength" : 255,
    "minLength" : 1,
    "pattern" : "^[0-9a-zA-Z\\.\\-]*$",
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 560,
    "serviceID" : 2004
  }, {
    "instanceID" : 2013,
    "keyName" : "common.remoteCommand",
    "displayName" : "Command",
    "defaultValue" : "",
    "value" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Specify the full path of the command to be executed on the
execution target server. If the path contains a space, enclose the entire path in
double quotation marks.",
    "mode" : "in",
    "required" : true,
    "maxLength" : 256,
    "minLength" : 1,
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 560,
    "serviceID" : 2004
  }, {
    "instanceID" : 2017,
```

```

    "keyName" : "common.remoteCommandParameter",
    "displayName" : "Command parameters",
    "defaultValue" : "",
    "value" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "Specify the parameters for the command to be executed on the
execution target server. If a parameter contains a space, enclose the entire
parameter in double quotation marks.",
    "mode" : "in",
    "required" : false,
    "maxLength" : 1024,
    "minLength" : 1,
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 560,
    "serviceID" : 2004
  }, {
    "instanceID" : 2016,
    "keyName" : "common.stdoutProperty",
    "displayName" : "Standard output string",
    "defaultValue" : "",
    "value" : "",
    "type" : "string",
    "visibility" : "exec",
    "scope" : "local",
    "description" : "This property contains the character string output to standard
output by the specified command. ",
    "mode" : "out",
    "required" : false,
    "propertyGroupName" : "reserved.defaultGroup",
    "validationScript" : "",
    "readOnly" : false,
    "hidden" : false,
    "reference" : false,
    "serviceTemplateID" : 560,
    "serviceID" : 2004
  } ],
  "count" : 4
}

```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.9.5 Acquisition of a list of property values

### Function

Acquires a list of the following values:

- Service share properties
- Properties related to a specific service

- Properties related to a specific schedule
- Properties related to a specific task

## Execution permissions

Admin role, Develop role, Modify role, Submit role

## API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/PropertyValues
```

This API acquires a list of property values for all services, schedules, and tasks, for which the user who executed the API has permissions. By specifying query parameters, you can filter the property values for which you want to acquire the list. Specify query parameters in the following format:

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-76: Query parameters that can be set for API Acquisition of a list of property values

Query parameter	Filter condition
serviceID	Equal to the specified value
scheduleID	
taskID	

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

### Example

The following example specifies 16731 for serviceID and 512 for taskID as query parameters.

```
?serviceID=16731&taskID=512
```

If you want to acquire the property value for a service, schedule, or task, you need to specify a query parameter for the corresponding serviceID, scheduleID, or taskID. If no query parameter is specified, only service share properties are returned as the response.

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

Data that matches the conditions specified by query parameters in a request is returned in the response body. The following shows the schema of the response body.

```
{
  "data" : [ {"member-of-the-resources-for-property-value-management-
functionality(PropertyValues)" : value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API acquires a list of property values for all services, schedules, and tasks.

Request header:

```
GET /Automation/v1/objects/PropertyValues HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:40:06 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO 56ca4c95167e4ce4aeb51fa73a85b2923d65e28e_V0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "instanceID" : 25,
    "type" : "boolean",
    "keyName" : "com.hitachi.software.dna.sys.mail.notify",
    "value" : "false",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 24,
    "type" : "string",
    "keyName" : "com.hitachi.software.dna.sys.mail.smtp.server",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 18,
    "type" : "integer",
    "keyName" : "com.hitachi.software.dna.sys.mail.smtp.port",
    "value" : "25",
    "readOnly" : false,
```

```

    "hidden" : false
  }, {
    "instanceID" : 5,
    "type" : "string",
    "keyName" : "com.hitachi.software.dna.sys.mail.smtp.userid",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 14,
    "type" : "password",
    "keyName" : "com.hitachi.software.dna.sys.mail.smtp.password",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 9,
    "type" : "string",
    "keyName" : "com.hitachi.software.dna.sys.mail.from",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 20,
    "type" : "string",
    "keyName" : "com.hitachi.software.dna.sys.mail.to",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 28,
    "type" : "string",
    "keyName" : "com.hitachi.software.dna.sys.mail.cc",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 21,
    "type" : "string",
    "keyName" : "com.hitachi.software.dna.sys.mail.bcc",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  } ],
  "count" : 9
}

```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.9.6 Batch update of property values

### Function

Updates the following property values in a batch:

- Property values related to specific tasks
- Property values related to specific services

- Service share property values
- Property values for multiple services

## Execution permissions

Admin role, Develop role, Modify role

## API version

v1

## Request format

```
PUT http://host:port/Automation/version/objects/PropertyValues
```

The following shows the structure of the request body.

```
{
  "pagination" : { },
  "data" : [...],
  "count" : X#
}
```

#

*X* is replaced with a number.

The following table describes the object that can be specified as *data* (member) in the schema of a request.

Table 2-77: Object that can be specified as data (member)

Function	Resource name	Number	Description
Property value	PropertyValue	1	PropertyValue resource to be updated

The following table describes the properties that must be specified for this object.

Resource name	Member name	Number
PropertyValue	instanceID	0 to n
	value	
	readOnly	
	hidden	

If you do not specify query parameters, service share properties are updated in a batch.

By specifying the `serviceID` query parameter, you can filter the property values to be updated in a batch. The attributes `readOnly` and `hidden` can be updated when you specify query parameters. Specify query parameters in the following format:

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-78: Query parameter that can be set for the API function Batch update of property values

Query parameter	Filter condition
serviceID	Equal to the specified value

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	The property value is invalid, or the resource cannot be edited.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have update permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "data" : [ {"member-of-the-resources-for-property-value-
functionality(PropertyValues)" : value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API function updates service share properties in a batch.

Request header:

```
PUT /Automation/v1/objects/PropertyValues HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:40:16 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
1aa95d66e62d885b5583da3620bd166fd3a3_V1o8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
```



Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS  
Access-Control-Allow-Credentials: true  
Cache-Control: no-cache  
Transfer-Encoding: chunked  
Content-Type: application/json

Response body:

```
{
  "data" : [ {
    "instanceID" : 25,
    "type" : "boolean",
    "keyName" : "com.hitachi.software.dna.sys.mail.notify",
    "value" : "false",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 24,
    "type" : "string",
    "keyName" : "com.hitachi.software.dna.sys.mail.smtp.server",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 18,
    "type" : "integer",
    "keyName" : "com.hitachi.software.dna.sys.mail.smtp.port",
    "value" : "25",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 5,
    "type" : "string",
    "keyName" : "com.hitachi.software.dna.sys.mail.smtp.userid",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 14,
    "type" : "password",
    "keyName" : "com.hitachi.software.dna.sys.mail.smtp.password",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 9,
    "type" : "string",
    "keyName" : "com.hitachi.software.dna.sys.mail.from",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 20,
    "type" : "string",
    "keyName" : "com.hitachi.software.dna.sys.mail.to",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }, {
    "instanceID" : 28,
    "type" : "string",
    "keyName" : "com.hitachi.software.dna.sys.mail.cc",
    "value" : "",
    "readOnly" : false,
    "hidden" : false
  }
]
```

```

    }, {
      "instanceID" : 21,
      "type" : "string",
      "keyName" : "com.hitachi.software.dna.sys.mail.bcc",
      "value" : "",
      "readOnly" : false,
      "hidden" : false
    } ],
    "count" : 9
  }
}

```

## 2.9.7 Acquisition of a property value

### Function

Acquires information about the specified property value.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/PropertyValues/id
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The requested resource or operation does not exist, or the user does not have read permission for the resource.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```

{
  "instanceID" : instance-id,
  "type" : "type",
  "keyName" : "key-name",
  "value" : "value",
  "readOnly" : {true|false},

```

```
"hidden" : {true|false},
"serviceID" : service-id,
"scheduleID" : schedule-id,
"taskID" : task-id
}
```

## Usage example

In the following example, the API acquires information about the property value whose instanceID is 7.

Request header:

```
GET /Automation/v1/objects/PropertyValues/7 HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzZGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:40:54 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
c9f651825563b97bf5d72fea6b1b1cde07a3f41_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : 7,
  "type" : "string",
  "keyName" : "com.hitachi.software.dna.sys.mail.cc",
  "value" : "",
  "readOnly" : false,
  "hidden" : false
}
```

## 2.9.8 Update of a property value

### Function

Updates the property value that has the specified ID.

### Execution permissions

Admin role, Develop role, Modify role

## API version

v1

## Request format

```
PUT http://host:port/Automation/version/objects/PropertyValues/id
```

The request schema has the same format as the response body for the API function Acquisition of a property value. The following table describes the object that can be specified as *PropertyValue* (member).

Table 2-79: Object that can be specified as PropertyValue (member)

Function	Resource name	Number	Description
Property value	PropertyValue	1	PropertyValue resource that has the specified ID

The following table describes the property that must be specified for this object.

Resource name	Member name	Number
PropertyValue	value	1

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	The property value is invalid, or the resource cannot be edited.
401	Unauthorized	The user does not have login permission.
403	Forbidden	The user does not have update permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : instance-id,
  "type" : "type",
  "keyName" : "key-name",
  "value" : "value",
  "readOnly" : {true|false},
  "hidden" : {true|false},
  "serviceID" : service-id,
  "scheduleID" : schedule-id,
  "taskID" : task-id
}
```

## Usage example

In the following example, the API function updates the value of the property whose instanceID is 24.

Request header:

```
PUT /Automation/v1/objects/PropertyValues/24 HTTP/1.1
Host: 10.196.184.238:22015
User-Agent: useragent1
Accept: application/json
Content-Type: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:40:59 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
d37375c943b0fcff62645a210ed9a96d116e153_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : 24,
  "type" : "string",
  "keyName" : "com.hitachi.software.dna.sys.mail.smtp.server",
  "value" : "server",
  "readOnly" : false,
  "hidden" : false
}
```

## 2.9.9 Acquisition of a list of operations for a property value

### Function

Acquires a list of operations that can be executed for the specified property value.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/PropertyValues/id/actions
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "data" : [ {"member-of-the-resources-for-property-value-
functionality(PropertyValues)" : value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

## Usage example

In the following example, the API acquires a list of operations that can be executed for the property value whose instanceID is 9.

Request header:

```
GET /Automation/v1/objects/PropertyValues/9/actions HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzZGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2014 12:41:31 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
1aa95d66e62d885b5583da3620bd166fd3a3_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "name" : "update",
    "href" : "http://10.196.184.238:22015/Automation/v1/objects/PropertyValues/9",
    "method" : "PUT",
    "parameters" : [ ]
  } ],
  "count" : 1
}
```

## 2.9.10 Acquisition of a list of property groups

### Function

Acquires a list of property groups that the properties retained by a service belong to.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/PropertyGroups
```

When you execute the API function, make sure that you specify a query parameter to filter property groups for which you want to acquire the list. You cannot specify multiple query parameters. Specify a query parameter in the following format:

```
?query-parameter=value
```

Table 2-80: List of query parameters that can be specified for the API function Acquisition of a list of property groups

Query parameter	Filter condition
serviceTemplateID	Equal to the specified value
serviceID	
scheduleID	
taskID	

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

Data that matches the conditions specified by a query parameter in a request is returned in the response body. The following shows the schema of the response body.

```
{
  "data" : [ {"member-of-the-resources-for-property-group-
functionality(PropertyGroups)" : value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-the-query-
parameter(0-to-n)
}
```

## Usage example

In the following example, the API function acquires a list of property groups that the properties retained by the service whose serviceID is 3134 belong to.

Request header:

```
GET /Automation/v1/objects/PropertyGroups?serviceID=3134 HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 03 Aug 2015 04:06:07 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
b3e9a4ed913c5b5bc941f48bfb1333ced0f1fff6_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "keyName" : "reserved.defaultGroup",
    "displayName" : "reserved.defaultGroup",
    "description" : "",
    "ordinal" : 0,
    "validationScript" : "",
```



```
    "display" : "config,submit,taskDetail"  
  } ],  
  "count" : 1  
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
-

## 2.10 Service group-related API functions

### 2.10.1 Acquisition of a list of service groups

#### Function

Acquires a list of service groups.

#### Execution permissions

Admin role, Develop role, Modify role, Submit role

#### API version

v1

#### Request format

```
GET http://host:port/Automation/version/objects/ServiceGroups
```

#### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
400	Bad Request	A query parameter is invalid.
401	Unauthorized	The user does not have login permission.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

#### Response schema

Data that matches the conditions specified by query parameters in a request is returned in the response body. The following shows the schema of the response body.

```
{
  "data" : [ {"member-of-the-resources-for-service-group-
functionality(ServiceGroups)" : value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

#### Usage example

In the following example, the API function acquires a list of service groups.

```
Request header:
```

```
GET /Automation/v1/objects/ServiceGroups HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 07:09:41 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
75f7726f932537efbc38f15ea81c31a8797bable_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "instanceID" : 3,
    "objectID" : "Automation_RG_DEFAULT",
    "name" : "DefaultServiceGroup",
    "description" : "default service group"
  }, {
    "instanceID" : 2,
    "objectID" : "Automation_RG_ALL",
    "name" : "All Service Groups",
    "description" : "default service groups which contains all services"
  } ],
  "count" : 2
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.10.2 Acquisition of information about a service group

### Function

Acquires information about the specified service group.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/ServiceGroups/id
```

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

The following shows the structure of the response body for a request.

```
{
  "instanceID" : instance-id,
  "objectID" : "object-id"
  "name" : "name",
  "description" : "description"
}
```

## Usage example

In the following example, the API function acquires information about the service group whose instanceID is 3.

Request header:

```
GET /Automation/v1/objects/ServiceGroups/3 HTTP/1.1
Host: 192.168.146.132:22015
User-Agent: curl/7.36.0
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 07:11:12 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
1f2d33f62adb5df5ca712acb2a0a430cb986e_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "instanceID" : 3,
  "objectID" : "Automation_RG_DEFAULT",
  "name" : "DefaultServiceGroup",
  "description" : "default service group"
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.10.3 Acquisition of a list of operations for a service group

### Function

Acquires a list of operations that can be executed for the specified service group.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/objects/ServiceGroups/id/actions
```

### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
404	Not found	The permission is invalid, or the resource does not exist.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

### Response schema

The following shows the structure of the response body for a request.

```
{
  "data" : [ ],
  "count" : 0
}
```

## Usage example

In the following example, the API function acquires a list of operations that can be performed for the service group whose instanceID is 5186.

Request header:

```
GET /Automation/v1/objects/ServiceGroups/5186/actions HTTP/1.1
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
Host: 192.168.146.132:22015
Accept: application/json
User-Agent: curl/7.36.0
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 07:16:43 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
d5802c6c6df5bf91a24f7f372be1af96a241eae_Vlo8Y30JBWoKHUYTEXAMXx5iHgQ=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ ],
  "count" : 0
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
-

## 2.11 Tag-related API functions

### 2.11.1 Acquisition of a list of tag groups

#### Function

Acquires a list of tag groups. In addition, this API function acquires a list of tags that belong to each tag group.

#### Execution permissions

Admin role, Develop role, Modify role, Submit role

#### API version

v1

#### Request format

```
GET http://host:port/Automation/version/objects/TagGroups
```

#### Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

#### Response schema

The following shows the structure of the response body for a request.

```
{
  "data" : [ {"member-of-the-resources-for-tag-group-functionality(TagGroups)" :
value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
parameters (0-to-n)
}
```

#### Usage example

In the following example, the API function acquires a list of tag groups.

Request header:

```
GET /Automation/v1/objects/TagGroups HTTP/1.1
Authorization: Basic c3lzdGVtOm1hbmFnZXI=
```

```
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 03:37:17 GMT
Server Cosminexus HTTP Server is not blacklisted
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
d72a9887e1aef533d4763b1adf0a391d6cfa6cb_Vlo8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "instanceID" : 34,
    "name" : "Applications",
    "tags" : "SQL Server,XenDesktop,Oracle Database,Cluster,Exchange"
  }, {
    "instanceID" : 42,
    "name" : "Hypervisors",
    "tags" : "VMware vSphere,Hyper-V"
  }, {
    "instanceID" : 45,
    "name" : "Storage Services",
    "tags" : "Replicate Storage,Add Like Storage,Snapshot,Add New Storage"
  }, {
    "instanceID" : 54,
    "name" : "Uncategorized",
    "tags" : "Basic,Hitachi,Windows,Linux"
  } ],
  "count" : 4
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
- 

## 2.11.2 Acquisition of a list of tags

### Function

Acquires a list of tags that are set for the specified resource.

### Execution permissions

Admin role, Develop role, Modify role, Submit role



## API version

v1

## Request format

```
GET http://host:port/Automation/version/objects/Tags
```

This API function acquires a list of all tags for which the user who executed the API function has permissions. By specifying query parameters, you can filter the tags for which you want to acquire the list. Specify query parameters in the following format:

```
?query-parameter=value[&query-parameter=value...]
```

Table 2-81: List of query parameters that can be specified for the API function Acquisition of a list of tags

Query parameter		Filter condition
detail		Acquires details of tags.
resourceType <sup>#1</sup>	ServiceTemplate	Equal to the specified value
	Service	
	Task	
	TaskHistory	

#1

If you specify resourceType, query parameters and HQL::filter are valid for the resource specified for resourceType.

For details about other query parameters that can be specified, see [2.2.9 Query parameter](#).

## Status code

The following table describes the various status codes that can be returned as the response to a request.

Status code	Message	Description
200	OK	Processing has been successfully completed.
401	Unauthorized	The user does not have login permission.
412	Precondition failed	The server is not available.
500	Server-side error	A server processing error occurred.

## Response schema

Data that matches the conditions specified by query parameters in a request is returned in the response body. The following shows the schema of the response body.

```
{
  "data" : [ {"member-of-the-resources-for-tag-functionality-(Tags)" :
value ... }, ... ],
  "count" : number-of-data-items-that-match-the-conditions-specified-by-query-
```

```
parameters (0-to-n)
}
```

## Usage example

In the following example, the API function acquires a list of all tags.

Request header:

```
GET /Automation/v1/objects/Tags HTTP/1.1
Authorization: Basic c3lzZGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2015 03:38:52 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
d2729dff1c31a47ed713d92612eec93fe7919c8_Vlo8Y30JdDBUB3ljJSVParTjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "data" : [ {
    "instanceID" : 35,
    "name" : "Replicate Storage",
    "tagGroupID" : 45
  }, {
    "instanceID" : 36,
    "name" : "SQL Server",
    "tagGroupID" : 34
  }, {
    "instanceID" : 37,
    "name" : "Add Like Storage",
    "tagGroupID" : 45
  }, {
    "instanceID" : 38,
    "name" : "Snapshot",
    "tagGroupID" : 45
  }, {
    "instanceID" : 39,
    "name" : "Add New Storage",
    "tagGroupID" : 45
  }, {
    "instanceID" : 40,
    "name" : "VMware vSphere",
    "tagGroupID" : 42
  }, {
    "instanceID" : 41,
    "name" : "XenDesktop",
    "tagGroupID" : 34
  }, {
```

```
    "instanceID" : 43,
    "name" : "Hyper-V",
    "tagGroupID" : 42
  }, {
    "instanceID" : 44,
    "name" : "Oracle Database",
    "tagGroupID" : 34
  }, {
    "instanceID" : 46,
    "name" : "Cluster",
    "tagGroupID" : 34
  }, {
    "instanceID" : 47,
    "name" : "Exchange",
    "tagGroupID" : 34
  }, {
    "instanceID" : 51,
    "name" : "Basic",
    "tagGroupID" : 54
  }, {
    "instanceID" : 52,
    "name" : "Hitachi",
    "tagGroupID" : 54
  }, {
    "instanceID" : 552,
    "name" : "Windows",
    "tagGroupID" : 54
  }, {
    "instanceID" : 559,
    "name" : "Linux",
    "tagGroupID" : 54
  }, {
    "instanceID" : 564,
    "name" : "Execute Script",
    "tagGroupID" : 54
  }, {
    "instanceID" : 1004,
    "name" : "Report Volume Information to Replication Manager",
    "tagGroupID" : 54
  } ],
  "count" : 17
}
```

---

## Related topics

- [2.2.14 Members of resources](#)
-

## 2.12 API functions for information management

---

This section describes the operations for acquiring user information, or JPI/AO and API version information.

### 2.12.1 Acquisition of user information

#### Function

Acquires information about users that execute API functions.

#### Execution permissions

Admin role, Develop role, Modify role, Submit role

#### API version

v1

#### Request format

```
GET http://host:port/Automation/version/user
```

#### Status code

For details about the status codes that can be returned as the response to a request, see the relevant topic in [2.2.17 Status code](#).

#### Response schema

The following shows the structure of the response body for a request.

```
{
  "userName" : "user-name",
  "accessPermission" : [ ... ],
  "fullName" : "full-name",
  "description" : "description",
  "email" : "email",
  "resourceGroup" : [ {
    "instanceId" : "instance-id",
    "name" : "resource-group-name",
    "description" : "description",
    "accessPermission" : [ ... ]
  } ]
}
```

#### Usage example

In the following example, the API acquires information about the execution user.

```
Request header:
GET /Automation/v1/user HTTP/1.1
```

```
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: curl/7.36.0
Host: 10.196.184.182:22015
Accept: application/json
```

Response header:

```
HTTP/1.1 200 OK
Date: Thu, 30 Jul 2015 07:17:47 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO
31fd21f2412025969969b479f296b5be20b267_V1o8Y30JdDBUB3ljJSVPaRtjBSA=_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-store, no-transform
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "userName" : "System",
  "accessPermission" : [ "User Management" ],
  "fullName" : "",
  "description" : "Built-in account",
  "email" : "",
  "resourceGroup" : [ {
    "instanceID" : "Automation_RG_ALL",
    "name" : "All Service Groups",
    "description" : "default service groups which contains all services",
    "accessPermission" : [ "View", "Execute", "Develop", "Modify", "Admin" ]
  } ]
}
```

## 2.12.2 Acquisition of version information

### Function

Acquires the JP1/AO and API version.

### Execution permissions

Admin role, Develop role, Modify role, Submit role

### API version

v1

### Request format

```
GET http://host:port/Automation/version/configuration/version
```

## Status code

For details about the status codes that can be returned as the response to a request, see the relevant topic in [2.2.17 Status code](#).

## Response schema

The following shows the structure of the response body for a request.

```
{
  "productName" : "product-name",
  "productVersion" : "product-version",
  "apiVersion" : "api-version"
}
```

## Usage example

In the following example, the API acquires version information.

Request header:

```
GET /Automation/v1/configuration/version HTTP/1.1
Host: 10.196.184.238:22015
Accept: application/json
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1
```

Response header:

```
HTTP/1.1 200 OK
Date: Mon, 28 Jul 2014 04:34:59 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO 4e671d509ad3cd624d83afd9da20f55c1c261193_WIN-JLTV0PQLK2A_V0810
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

Response body:

```
{
  "productName" : "JP1/Automatic Operation",
  "productVersion" : "11-00-00",
  "apiVersion" : "01.01.00"
}
```

## 2.13 API usage example

The procedure below shows an example of using an API function to execute a service. First, check the instanceID of the service you want to execute. Then, specify `immediate` for the schedule type and execute the service.

1. Display a list of resources for service functionality (Services), and check the instanceID of the service you want to execute.

```
GET /Automation/v1/objects/Services/ HTTP/1.1
Host:10.197.112.78:22015
User-Agent:sample rest client/1.00.0
Accept:application/json
Accept-Language: ja
Authorization: Basic c3lzdGVtOmlhbmFnZXI=
User-Agent: useragent1

HTTP/1.1 200 OK
Date: Mon, 18 Aug 2014 11:15:01 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO b8712c86fcd026562182a358ea43bb23b09c62_V0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json

{
  "data" : [ {
    "instanceID" : 2269, <- Check instanceID.
    "name" : "Execute remote command",
    "description" : "Executes a command on the remote execution target server.",
    "tags" : "OS_Operations,Basic",
    "serviceTemplateName" : "Execute remote command",
    "createTime" : "2014-08-18T16:53:50.000+0900",
    "modifyTime" : "2014-08-18T16:53:58.000+0900",
    "serviceState" : "release",
    "serviceGroupName" : "DefaultServiceGroup",
    "iconURL" : "http://10.197.112.78:22015/Automation/icon/services/
com.hitachi.software.dna.cts.jp1/remoteCommandExe/01.10.00",
    "vendorName" : "Hitachi,Ltd.",
    "version" : "01.10.00",
    "latest" : true,
    "imageURL" : "http://10.197.112.78:22015/Automation/resources/images/overview/
overview.png",
    "serviceTemplateID" : 2204,
    "serviceGroupID" : 2
  } ],
  "count" : 1
}
```

2. Acquire a list of operations that can be performed for the resource that has the instanceID you checked above.

```
GET /Automation/v1/objects/Services/2269/actions HTTP/1.1
Host:10.197.112.78:22015
User-Agent:sample rest client/1.00.0
Accept:application/json
Accept-Language: ja
Authorization: HSSO b8712c86fcd026562182a358ea43bb23b09c62_V0300

HTTP/1.1 200 OK
Date: Mon, 18 Aug 2014 11:24:41 GMT
```

```

Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO b365e6a2cda2b4d195d55fee1461a6ed0889927_V0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json

{
  "data" : [ {
    "name" : "submit", <- Check the href information of submit that is used to
execute the service.
    "href" : "http://10.197.112.78:22015/Automation/v1/objects/Services/2269/
actions/submit/invoke",
    "method" : "POST",
    "parameters" : [ ]
  } ],
  "count" : 1
}

```

### 3. Acquire the request body information that is necessary for the operation to execute the specified service.

```

GET /Automation/v1/objects/Services/2269/actions/submit HTTP/1.1
Host:10.197.112.78:22015
User-Agent:sample rest client/1.00.0
Accept:application/json
Accept-Language: ja
Authorization: HSSO b8712c86fcd026562182a358ea43bb23b09c62_V0300

HTTP/1.1 200 OK
Date: Mon, 18 Aug 2014 11:26:00 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO 1b9b5891c58315e26cd0cca9aac6d43e572e3db_V0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json

Output the response to the properties.json file.

```

#### Contents of properties.json

```

{
  "name" : "submit",
  "href" : "http://10.197.112.78:22015/Automation/v1/objects/Services/2269/
actions/submit/invoke",
  "method" : "POST",
  "parameters" : [ {
    "name" : "Execute remote command_20140818202600",
    "submitter" : "System",
    "scheduleType" : "immediate",
    "description" : "",
    "scheduledStartTime" : "2014-08-18T20:26:00.536+09:00",
    "recurrenceInterval" : "daily",
    "recurrenceDayOfWeek" : "",
    "recurrenceDayOfMonth" : "",
    "recurrenceLastDayOfMonth" : false,
    "recurrenceStartDate" : "2014-08-18",
    "recurrenceTime" : "00:00:00",

```



```

    "serviceID" : 2269
  }, {
    "instanceID" : 2275,
    "type" : "string",
    "keyName" : "common.targetHost",
    "value" : "",
    "serviceID" : 2269
  }, {
    "instanceID" : 2271,
    "type" : "string",
    "keyName" : "common.remoteCommand",
    "value" : "",
    "serviceID" : 2269
  }, {
    "instanceID" : 2273,
    "type" : "string",
    "keyName" : "common.remoteCommandParameter",
    "value" : "",
    "serviceID" : 2269
  } ]
}

```

4. Edit the acquired template information as necessary. The following is an example of specifying immediate for the schedule type.

```

{
  "name" : "submit",
  "href" : "http://10.197.112.78:22015/Automation/v1/objects/Services/2269/
actions/submit/invoke",
  "method" : "POST",
  "parameters" : [ {
    "name" : "Execute remote command_20140818202600",
    "submitter" : "System",
    "scheduleType" : "immediate", <- Specify "immediate".
    "description" : "",
    "scheduledStartTime" : "2014-08-18T20:26:00.536+09:00",
    "recurrenceInterval" : "daily",
    "recurrenceDayOfWeek" : "",
    "recurrenceDayOfMonth" : "",
    "recurrenceLastDayOfMonth" : false,
    "recurrenceStartDate" : "2014-08-18",
    "recurrenceTime" : "00:00:00",
    "serviceID" : 2269
  }, {
    "instanceID" : 2275,
    "type" : "string",
    "keyName" : "common.targetHost",
    "value" : "", <- Change value as necessary.
    "serviceID" : 2269
  }, {
    "instanceID" : 2271,
    "type" : "string",
    "keyName" : "common.remoteCommand",
    "value" : "hostname", <- Change value as necessary.
    "serviceID" : 2269
  }, {
    "instanceID" : 2273,
    "type" : "string",
    "keyName" : "common.remoteCommandParameter",
    "value" : "", <- Change value as necessary.
    "serviceID" : 2269
  } ]
}

```

## 5. Execute the service by using the edited information.

```
POST /Automation/v1/objects/Services/2269/actions/submit/invoke HTTP/1.1
Host:10.197.112.78:22015
User-Agent:sample rest client/1.00.0
Accept:application/json
Accept-Language: ja
Content-Type: application/json
Content-Length: 1087
Authorization: HSSO b8712c86fcd026562182a358ea43bb23b09c62_V0300
```

Request the contents of properties.json.

```
HTTP/1.1 200 OK
Date: Mon, 18 Aug 2014 11:39:03 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO 8ba382c1f2e81a65d7a252391b262624c6fa61_V0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

```
{
  "instanceID" : "4a9141e4-9566-4e42-af08-0f09926f2a5f",
  "created" : "2014-08-18T20:39:04.242+0900",
  "updated" : "2014-08-18T20:39:04.242+0900",
  "completed" : "2014-08-18T20:39:04.242+0900",
  "state" : "success",
  "affectedResource" : [ "http://10.197.112.78:22015/Automation/v1/objects/
Schedules/2285" <- URL of the created schedule resource,
"http://10.197.112.78:22015/Automation/v1/objects/Tasks/2280" <- URL of the
created task resource ],
  "result" : [ ]
}
```

## Acquire the Schedule resource created by execution of the service, and check the contents of the resource.

```
GET /Automation/v1/objects/Schedules/2285 HTTP/1.1
Host:10.197.112.78:22015
User-Agent:sample rest client/1.00.0
Accept:application/json
Accept-Language: ja
Authorization: HSSO b8712c86fcd026562182a358ea43bb23b09c62_V0300
```

```
HTTP/1.1 200 OK
Date: Mon, 18 Aug 2014 11:43:00 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO cafcefb87199122267f8ad33772555f9357c8a2_V0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json
```

```
{
  "instanceID" : 2285,
  "name" : "Execute remote command_20140818202600",
  "submitter" : "System",
  "scheduleType" : "immediate",
```

```

"createTime" : "2014-08-18T20:39:03.000+0900",
"modifyTime" : "2014-08-18T20:39:03.000+0900",
"description" : "",
"serviceState" : "release",
"serviceID" : 2269
}

```

6. Acquire the Task resource created by execution of the service, and check the contents of the resource.

```

GET /Automation/v1/objects/Tasks/2280 HTTP/1.1
Host:10.197.112.78:22015
User-Agent:sample rest client/1.00.0
Accept:application/json
Accept-Language: ja
Authorization: HSSO b8712c86fcd026562182a358ea43bb23b09c62_V0300

HTTP/1.1 200 OK
Date: Mon, 18 Aug 2014 11:43:59 GMT
Server: Cosminexus HTTP Server
Access-Control-Expose-Headers: WWW-Authenticate
WWW-Authenticate: HSSO 3713abcd1e99d1481c7b92cc9892a95d1a702d6_V0300
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, HEAD, OPTIONS
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: application/json

{
  "instanceID" : 2280,
  "name" : "Execute remote command_20140818202600",
  "status" : "failed",
  "startTime" : "2014-08-18T20:39:04.000+0900",
  "completionTime" : "2014-08-18T20:39:13.000+0900",
  "submitter" : "System",
  "submitTime" : "2014-08-18T20:39:03.000+0900",
  "modifyTime" : "2014-08-18T20:39:19.000+0900",
  "serviceState" : "release",
  "scheduleType" : "immediate",
  "description" : "",
  "serviceName" : "Execute remote command",
  "tags" : "",
  "serviceGroupName" : "DefaultServiceGroup",
  "serviceTemplateID" : 2204,
  "scheduleID" : 2285,
  "serviceGroupID" : 2,
  "serviceID" : 2269
}

```

# Appendix

## A. Reference Information

---

This appendix provides reference information about how to use JP1/AO.

### A.1 Version changes

#### (1) Changes in version 11-50

- JP1/AJS3 is no longer included in JP1/AO, and therefore the `stopcluster` command is no longer required. Accordingly, descriptions of this requirement were deleted.
- `MD5withRSA` was deleted from the signature algorithm that can be specified by using the `hcnds64ssltool` command with the `sigalg` option specified.
- The value 5 was deleted from the list of return values for the `hcnds64dbtrans` command executed with the `export` option specified.

#### (2) Changes in version 11-10

- JP1/AO no longer uses JP1/AJS3 as a task processing engine, and content indicating otherwise was deleted.
- The `setupcluster` and `restoresystem` commands no longer use the `jpluser` option, and content indicating otherwise was deleted.
- The periodic execution cycle for executing services and tasks can now be specified in hourly units, and a description of this was added.
- JP1/Base is no longer a prerequisite product for JP1/AO, and descriptions of this requirement were deleted.
- Connection Status and Connected Time were added to the items output for a connection-destination definition information file.
- Operations that can be performed for tasks can now be specified. Accordingly, Supported Action Type was added to the items output for a list of services and a list of service templates.
- Supported Action Type was added to the items output for a list of tasks and to the file output examples. The figure illustrating an output example of a list of tasks was also changed.
- SupportedActionType was added to the members that can be acquired by using the following API functions: "Acquisition of a list of service templates", "Acquisition of a list of services", and "Acquisition of a list of tasks".
- Supported Action Type was added to the usage examples of the following API functions: "Acquisition of a list of service templates", "Acquisition of a list of services", and "Acquisition of a list of tasks".
- Supported Action Type was added to the response schema and usage examples of the following API functions: "Acquisition of information about a service template", "Acquisition of service information", and "Acquisition of task information".
- Supported Action Type was added to the response schema and usage examples of the following API functions: "Editing a service" and "Editing a task".
- The periodic execution cycle for executing services and tasks can now be specified in hourly units. Accordingly, the member `recurrenceMinutes` was added to the table used for the periodic execution of related APIs.

### (3) Changes in version 11-01

- When using the `submittask` to submit services for recurring execution, execution on specified dates and at the end of the month can now be specified together.

### (4) Changes in version 11-00

#### (a) Changes from the manual (3021-3-088-20)

- The following OSs were added to the supported OSs:
  - Linux 7
  - Oracle Linux 6 (x64)
  - Oracle Linux 7
  - CentOS 6 (x64)
  - CentOS 7
  - SUSE Linux 12
- The following OSs were deleted from the supported OSs:
  - Linux 5 (AMD/Intel 64)
  - Linux 5 Advanced Platform (AMD/Intel 64)
- Windows was migrated from the 32-bit version to the 64-bit version.
- The installation folders of JP1/AO for Windows and Common Component were changed.
- A description for when JP1/AO is used in an English or Chinese environment was added.
- The port number used between JP1/AO and a web browser was changed.
- JP1/AJS3 and JP1/AO whose versions are 11 can now coexist.
- For the names of the commands that can be executed in Windows, the `hcnds` part was changed to `hcnds64`.
- The `deleteremoteconnection` command that deletes a connection destination definition registered in JP1/AO was added.
- The `listremoteconnections` command that outputs a list of connection destination definitions registered in JP1/AO was added.
- The `setremoteconnection` command that adds or updates a connection destination definition was added.
- The items that can be output by the `listservices` command were changed.
- The items that can be output by the `listtasks` command were changed.
- The following API functions were added:  
Service template-related API functions
  - Acquisition of a list of service templates
  - Acquisition of information about a service template
  - Deletion of a service template
  - Acquisition of a list of operations for a service template
  - Acquisition of the HTML file necessary for importing a service template
  - Import of a service template
  - Acquisition of information necessary for exporting a service template

- Export of a service template
- Acquisition of the URL for displaying the details of a service template
- Acquisition of information necessary for creating a service based on a service template
- Creation of a service based on a service template

#### Service-related API functions

- Editing a service
- Deletion of a service
- Acquisition of information necessary for resetting the counter for a service
- Reset of the counter for a service
- Acquisition of information necessary for the operation to change the status of a service to release
- Change of the status of a service to release
- Acquisition of information necessary for the operation to change the status of a service to maintenance
- Change of the status of a service to maintenance
- Acquisition of information necessary for the operation to change the status of a service to disabled
- Change of the status of a service to disabled
- Acquisition of the URL for the details of a service
- Acquisition of information necessary for changing the version of the service template used by a service
- Change of the version of the service template used by a service

#### Task-related API functions

- Editing a task
- Deletion of a task
- Acquisition of information necessary for forcibly stopping a task
- Forced stoppage of a task
- Acquisition of information necessary for responding to a task that is in the status Waiting for Response
- Response to a task that is in the status Waiting for Response
- Acquisition of information necessary for archiving a task
- Archiving a task

#### List of history-related API functions

- Acquisition of a list of history records
- Deletion of history records (with conditions specified)
- Acquisition of a history record
- Deletion of history records (with an ID specified)
- Acquisition of a list of operations for a history record

#### Property-related API functions

- Acquisition of lists of property definitions and property values
- Batch update of property values
- Update of a property value
- Acquisition of a list of property groups

### Service group-related API functions

- Acquisition of a list of service groups
- Acquisition of information about a service group
- Acquisition of a list of operations for a service group

### Tag-related API functions

- Acquisition of a list of tag groups
  - Acquisition of a list of tags
- According to the addition of the API functions, the following items were added or changed:
    - Domain names and resources that can be managed by API functions
    - Query parameter
    - Request header
    - Response header
    - Members of resources
    - Members to be returned for API functions that execute JP1/AO operations
    - Members to be returned for API functions that acquire executable operations
  - Descriptions of the status codes were added.

## **(b) Changes from the manual (3021-3-366(E))**

- Linux was added to the supported OSs.
- The installation folders of JP1/AO for Windows and Common Component were changed.
- The port number used between JP1/AO and a web browser was changed.
- Windows was migrated from the 32-bit version to the 64-bit version.
- JP1/AJS3 and JP1/AO whose versions are 11 can now coexist.
- For the names of the commands that can be executed in Windows, the `hcnds` part was changed to `hcnds64`.
- The `deleteremoteconnection` command that deletes a connection destination definition registered in JP1/AO was added.
- The `listremoteconnections` command that outputs a list of connection destination definitions registered in JP1/AO was added.
- The `setremoteconnection` command that adds or updates a connection destination definition was added.
- The items that can be output by the `listservices` command were changed.
- The items that can be output by the `listtasks` command were changed.
- A member that can be acquired by the operation Acquisition of a list of tasks or Acquisition of a list of steps was changed from `endTime` to `completionTime`.
- The following API functions were added:

### Service template-related API functions

- Acquisition of a list of service templates
- Acquisition of information about a service template
- Deletion of a service template



- Acquisition of a list of operations for a service template
- Acquisition of the HTML file necessary for importing a service template
- Import of a service template
- Acquisition of information necessary for exporting a service template
- Export of a service template
- Acquisition of the URL for displaying the details of a service template
- Acquisition of information necessary for creating a service based on a service template
- Creation of a service based on a service template

#### Service-related API functions

- Editing a service
- Deletion of a service
- Acquisition of information necessary for resetting the counter for a service
- Reset of the counter for a service
- Acquisition of information necessary for the operation to change the status of a service to release
- Change of the status of a service to release
- Acquisition of information necessary for the operation to change the status of a service to maintenance
- Change of the status of a service to maintenance
- Acquisition of information necessary for the operation to change the status of a service to disabled
- Change of the status of a service to disabled
- Acquisition of the URL for the details of a service
- Acquisition of information necessary for changing the version of the service template used by a service
- Change of the version of the service template used by a service

#### Task-related API functions

- Editing a task
- Deletion of a task
- Acquisition of information necessary for forcibly stopping a task
- Forced stoppage of a task
- Acquisition of information necessary for responding to a task that is in the status Waiting for Response
- Response to a task that is in the status Waiting for Response
- Acquisition of information necessary for archiving a task
- Archiving a task

#### List of history-related API functions

- Acquisition of a list of history records
- Deletion of history records (with conditions specified)
- Acquisition of a history record
- Deletion of history records (with an ID specified)
- Acquisition of a list of operations for a history record

#### Property-related API functions

- Acquisition of lists of property definitions and property values
- Batch update of property values
- Update of a property value
- Acquisition of a list of property groups

#### Service group-related API functions

- Acquisition of a list of service groups
- Acquisition of information about a service group
- Acquisition of a list of operations for a service group

#### Tag-related API functions

- Acquisition of a list of tag groups
- Acquisition of a list of tags

- According to the addition of the API functions, the following items were added or changed:
  - Domain names and resources that can be managed by API functions
  - Query parameter
  - Request header
  - Response header
  - Members of resources
  - Members to be returned for API functions that execute JPI/AO operations
  - Members to be returned for API functions that acquire executable operations
- A description stating that the `hcmds64getlogs` command can be executed even on the standby server of a cluster environment was added.
- A description stating that the `hcmds64getlogs` command can be executed even if the JPI/AO server is stopped was added.

## (5) Changes in version 10-54

### (a) Changes in the manual (3021-3-088-20)

- A member that can be acquired by the operation Acquisition of a list of tasks or Acquisition of a list of steps was changed from `endTime` to `completionTime`.

## (6) Changes in version 10-52

### (a) Changes in the manual (3021-3-088-10)

- Linux was added to the supported OSs.
- According to the addition of the function that manages plug-in versions, the following windows were added:
  - **Plug-in Version Management** dialog box (**Apply to All** tab)
  - **Plug-in Version Management** dialog box (**Individual apply** tab)
- A description of how to take action if a message dialog box indicating an unexpected error appears or windows are not displayed correctly when you log in to JPI/AO was added.

- Keyboard interactive authentication is now supported as an authentication method that can be used for SSH connection with operation-target devices.
- A description stating that the **Required** check box cannot be edited if you select the reserved plug-in property `plugin.publicKeyAuthentication` or `plugin.keyboardInteractiveAuthentication` was added.
- A description stating that the `hcmdsgetlogs` or `hcmds64getlogs` command can be executed even on the standby server of a cluster environment was added.
- A description stating that the `hcmdsgetlogs` or `hcmds64getlogs` command can be executed even if the JP1/AO server is stopped was added.

## (7) Changes in version 10-50

### (a) Changes in the manual (3021-3-088)

- A function that links with Active Directory to manage users was added.
- HTTPS connections are now supported.
- Public key authentication is now supported as a method of authenticating operation-target devices.
- The `stopcluster` command was added.  
Preparations for stopping JP1/AO services in a cluster environment can now be performed.
- The `hcmdsldapuser` command was added.  
The user information that is necessary to search Active Directory registration information when Active Directory linkage is used can now be edited.
- The `hcmdsssltool` command was added.  
A private key, CSR, self-signed certificate, and a file to contain the self-signed certificate, which are necessary for SSL connections, can now be created.
- A description stating that the following files are not targets of the `backupsystem` and `restoresystem` commands was added:
  - SSL server certificate files for HTTPS connections
  - Private key files for HTTPS connections
  - Private key files for public key authentication
- API functions are now supported.

### (b) Changes in the manual (3021-3-366(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 to the supported OSs.
- With addition of the task monitor function and the service template debugger function, the following windows were added:
  - **Task log** dialog box
  - **Debug-Tasks** view

- **Task Monitor** view
- **Perform Debugging** dialog box
- **Debug** view
- Service template debugging view
- A function that links with Active Directory to manage users was added.
- HTTPS connections are now supported.
- With the change to slide bars, the screenshots of the following windows were changed:
  - Main window
  - **Services** dialog box
  - **Tasks** window
  - **Tasks** view
  - **Task Histories** view
  - **Administration** window
  - **Connection Destinations** view
  - **Service Share Properties** view
  - **User Groups** view (**User Groups** tab)
  - **User Groups** view (**Users** tab)
  - **Resource Groups** view
  - **Editor** window
  - **Service template view** dialog box
  - Service template editing view
- A function for viewing the task log was added to the following windows:
  - **Waiting for Response Task List** dialog box
  - **In Progress Task List** dialog box
  - **Completed Task List** dialog box
  - **Failed Task List** dialog box
  - **Tasks** view
  - **Task Details** dialog box
- A function that retries tasks and a function that forcibly stops tasks were added to the following windows:
  - **Waiting for Response Task List** dialog box
  - **In Progress Task List** dialog box
  - **Tasks** view
- With the addition of the task monitor and service template debugger, the conditions in which the following windows are displayed were changed:
  - **Submit Service** dialog box
  - **Task Details** dialog box
  - **Respond** dialog box
  - **Plug-in** view

- **View Service Definition** dialog box
- **Steps** dialog box
- Service template editing view
- **Build/Release Result** dialog box

The descriptions of items displayed in the following windows were changed:

- **Tasks** window
- **Task Details** dialog box
- **Editor** window
- Service template editing view
- **Build/Release Result** dialog box
- *Return Value* was added to the list of steps displayed in the **Task Details** dialog box. In addition, a description stating that the statuses displayed in the list of steps can be changed was added.
- Public key authentication is now supported as a method of authenticating operation-target devices.
- Functions that can be executed or specified as root were added to the following windows:
  - **Plug-in** view
  - **Create Plug-in** dialog box
  - **Edit Plug-in** dialog box
- A Release plug-in can now be deleted.
- Content plug-ins that execute commands or scripts are now supported in AIX, HP-UX, and Solaris, in addition to Windows and Linux.
- The `stopcluster` command was added.  
Preparations for stopping JP1/AO services in a cluster environment can now be performed.
- The `hcmdsldapuser` command was added.  
The user information that is necessary to search Active Directory registration information when Active Directory linkage is used can now be edited.
- The `hcmdsssltool` command was added.  
A private key, CSR, self-signed certificate, and a file to contain the self-signed certificate, which are necessary for SSL connections, can now be created.
- The `listtasks` command can now be used to output the details of multiple tasks. In addition, the `submittask` command can now be used to re-register multiple tasks that are to be executed periodically or according to the schedule.
- A description of the `/user` option of the `hcmdscheckauth` command was added, and return value 247 was added.
- A note on the user name or password to be specified for an option was added.
- A description stating that debug services and debug tasks are not targets of the following commands was added:
  - `listservices` command
  - `listtasks` command
  - `stoptask` command
  - `submittask` command
- The `submittask` command can now be used to register a command that is to be executed periodically.

- A description stating that the following files are not targets of the `backupsystem` and `restoresystem` commands was added:
  - SSL server certificate files for HTTPS connections
  - Private key files for HTTPS connections
  - Private key files for public key authentication
- The description of the JP1/Base service was deleted because the JP1/Base service starts when the JP1/AO service starts.
- Notes that apply in a cluster system were added.
- The descriptions of the `/workpath` and `/file` options were changed. In addition, notes on the `hcmdsdbtrans` command were added.
- Explanations of debug services and debug tasks were added in the description of the `restoresystem` command. In addition, a description stating that the retry operation cannot be selected for restored tasks and debug tasks was added.
- API functions are now supported.
- Descriptions of the status icons displayed in windows were added.

## (8) Changes in version 10-12

### (a) Changes in the manual (3021-3-084-50)

- Windows Server 2012 R2 was added to the supported OSs.
- With addition of the task monitor function and the service template debugger function, the following windows were added:
  - **Task log** dialog box
  - **Debug-Tasks** view
  - **Task Monitor** view
  - **Perform Debugging** dialog box
  - **Debug** view
  - Service template debugging view
- With the change to slide bars, the screenshots of the following windows were changed:
  - Main window
  - **Services** dialog box
  - **Tasks** window
  - **Tasks** view
  - **Task Histories** view
  - **Administration** window
  - **Connection Destinations** view
  - **Service Share Properties** view
  - **User Groups** view (**User Groups** tab)
  - **User Groups** view (**Users** tab)
  - **Resource Groups** view

- **Editor** window
- **Service template view** dialog box
- Service template editing view
- A function for viewing the task log was added to the following windows:
  - **Waiting for Response Task List** dialog box
  - **In Progress Task List** dialog box
  - **Completed Task List** dialog box
  - **Failed Task List** dialog box
  - **Tasks** view
  - **Task Details** dialog box
- A function that retries tasks and a function that forcibly stops tasks were added to the following windows:
  - **Waiting for Response Task List** dialog box
  - **In Progress Task List** dialog box
  - **Tasks** view
- With the addition of the task monitor and service template debugger, the conditions in which the following windows are displayed were changed:
  - **Submit Service** dialog box
  - **Task Details** dialog box
  - **Respond** dialog box
  - **Plug-in** view
  - **View Service Definition** dialog box
  - **Steps** dialog box
  - Service template editing view
  - **Build/Release Result** dialog box

The descriptions of items displayed in the following windows were changed:

- **Tasks** window
- **Task Details** dialog box
- **Editor** window
- Service template editing view
- **Build/Release Result** dialog box
- *Return Value* was added to the list of steps displayed in the **Task Details** dialog box. In addition, a description stating that the statuses displayed in the list of steps can be changed was added.
- Functions that can be executed or specified as root were added to the following windows:
  - **Plug-in** view
  - **Create Plug-in** dialog box
  - **Edit Plug-in** dialog box
- A description stating that debug services and debug tasks are not targets of the following commands was added:
  - `listservices` command

- `listtasks` command
- `stoptask` command
- `submittask` command
- Explanations of debug services and debug tasks were added in the description of the `restoresystem` command. In addition, a description stating that the retry operation cannot be selected for restored tasks and debug tasks was added.
- Descriptions of the status icons displayed in windows were added.

## (9) Changes in version 10-11

### (a) Changes in the manual (3021-3-084-40)

- A Release plug-in can now be deleted.
- Content plug-ins that execute commands or scripts are now supported in AIX, HP-UX, and Solaris, in addition to Windows and Linux.
- A description of the `/user` option of the `hcmdscheckauth` command was added, and return value 247 was added.
- A note on the user name or password to be specified for an option was added.
- The `listtasks` command can now be used to output the details of multiple tasks. In addition, the `submittask` command can now be used to re-register multiple tasks that are to be executed periodically or according to the schedule.
- The `submittask` command can now be used to register a command that is to be executed periodically.
- The description of the JP1/Base service was deleted because the JP1/Base service starts when the JP1/AO service starts.
- Notes that apply in a cluster system were added.
- The descriptions of the `/workpath` and `/file` options were changed. In addition, notes on the `hcmdsdbtrans` command were added.

## (10) Changes in version 10-10

### (a) Changes in the manual (3021-3-084-30)

- New functionality allows the user to develop service templates and plug-ins in the **Editor** window.
- **Configuration Type** was added as a display item of the following windows:
  - **Waiting Task List** dialog box
  - **Waiting for Response Task List** dialog box
  - **In Progress Task List** dialog box
  - **Completed Task List** dialog box
  - **Failed Task List** dialog box
  - **Services** window
  - **Submit Service** dialog box
  - **Add Service** dialog box
  - **Service Definition** dialog box



- **Tasks** view
- **Task Details** dialog box
- **Task Histories** view
- The limit on simultaneous execution was increased from 2 to 10 for both the `submittask` and `stoptask` commands.
- The **Develop** role was added to user permissions. It can be used to execute the following commands:
  - `deleteservicetemplate` command
  - `importservicetemplate` command
  - `listservices` command
  - `listtasks` command
  - `stoptask` command
  - `submittask` command
- Configuration Type was added to the output items of the `listservices` and `listtasks` commands.
- The following description was added: If you omit specifying the `/property` option in the `submittask` command, the values you entered in the **Service Definition** dialog box are set for the corresponding property keys.
- The `/wait` option, which is used to finish the command after outputting the execution results of the task, was added to the `submittask` command.
- The `/scheduledate` and `/schedulesettime` options, which are used to specify when services are to be executed, were added to the `submittask` command.
- The limit values for the **Editor** window were added to the list of limit values.

## (b) Changes in the manual (3021-3-315-10(E))

- New functionality allows the user to develop service templates and plug-ins in the **Editor** window.
- Notes on operating on windows in Windows Server 2012 were added.
- **Configuration Type** was added as a display item of the following windows:
  - **Waiting Task List** dialog box
  - **Waiting for Response Task List** dialog box
  - **In Progress Task List** dialog box
  - **Completed Task List** dialog box
  - **Failed Task List** dialog box
  - **Services** window
  - **Submit Service** dialog box
  - **Add Service** dialog box
  - **Service Definition** dialog box
  - **Tasks** view
  - **Task Details** dialog box
  - **Task Histories** view
- The following items of the Task Details dialog box were changed:
  - **Jobnet Information** was changed to **Step Information**.

- **Jobnet Details** was changed to **Step Details**.
- **Root Jobnet Name** was deleted.
- **Units** was changed to **Steps**.
- **Unit Name** was changed to **Name**.
- **Comment** was changed to **Description**.
- Telnet was added to the available protocols.
- Plug-in resource files for English, Chinese, and Japanese environments can now be selected.
- Service resource files for English, Chinese, and Japanese environments can now be selected.
- The limit on simultaneous execution was increased from 2 to 10 for both the `submittask` and `stoptask` commands.
- New functionality allows the user to change the subject identification information output to the audit log.
- A description that the `setupcluster` command is not available in Windows Server 2012 was added.
- The Develop role was added to user permissions. It can be used to execute the following commands:
  - `deleteservicetemplate` command
  - `importservicetemplate` command
  - `listservices` command
  - `listtasks` command
  - `stoptask` command
  - `submittask` command
- Configuration Type was added to the output items of the `listservices` and `listtasks` commands.
- The following description was added: If you omit specifying the `/property` option in the `submittask` command, the values you entered in the **Service Definition** dialog box are set for the corresponding property keys.
- The `/wait` option, which is used to finish the command after outputting the execution results of the task, was added to the `submittask` command.
- The `/scheduledate` and `/schedulesettime` options, which are used to specify when services are to be executed, were added to the `submittask` command.
- The description of JP1/Base services was deleted because these services automatically start when JP1/AO services start.
- The limit values for the **Editor** window were added to the list of limit values.
- The list of limit values was modified.
- Items related to functionality in the list of limit values were moved to the Job Management Partner 1/Automatic Operation Overview and System Design Guide as List of limit values of functions.

## (11) Changes in version 10-02

### (a) Changes in the manual (3021-3-084-20)

- Notes on operating on windows in Windows Server 2012 were added.
- The following items of the Task Details dialog box were changed:
  - **Jobnet Information** was changed to **Step Information**.

- **Jobnet Details** was changed to **Step Details**.
- **Root Jobnet Name** was deleted.
- **Units** was changed to **Steps**.
- **Unit Name** was changed to **Name**.
- **Comment** was changed to **Description**.
- Telnet was added to the available protocols.
- New functionality allows the user to change the subject identification information output to the audit log.
- A description that the `setupcluster` command is not available in Windows Server 2012 was added.
- The list of limit values was modified.

## **(12) Changes in version 10-02**

### **(a) Changes in the manual (3021-3-084-10)**

- Items related to functionality in the list of limit values were moved to the Job Management Partner 1/Automatic Operation Overview and System Design Guide as List of limit values of functions.

# Index

## A

acquisition of history record 257  
acquisition of HTML file necessary for importing service template 142  
acquisition of information about service group 291  
acquisition of information about service template 136  
acquisition of information necessary for archiving task 241  
acquisition of information necessary for canceling schedule 196  
acquisition of information necessary for changing version of service template used by service 185  
acquisition of information necessary for creating service based on service template 150  
acquisition of information necessary for executing service 166  
acquisition of information necessary for exporting service template 146  
acquisition of information necessary for forcibly stopping task 222  
acquisition of information necessary for operation to change status of service to disabled 181  
acquisition of information necessary for operation to change status of service to maintenance 177  
acquisition of information necessary for operation to change status of service to release 174  
acquisition of information necessary for pausing schedule 200  
acquisition of information necessary for re-executing task 226  
acquisition of information necessary for resetting counter for service 171  
acquisition of information necessary for responding to task that is in the status Waiting for Response 230  
acquisition of information necessary for resuming schedule 203  
acquisition of information necessary for retrying task (retry from failed step) 234  
acquisition of information necessary for retrying task (retry from step after failed step) 238  
acquisition of information necessary for stopping task execution 219  
acquisition of list of histories 253  
acquisition of list of operations for history record 260  
acquisition of list of operations for property definition 271  
acquisition of list of operations for property value 285  
acquisition of list of operations for schedule 194

acquisition of list of operations for service 163  
acquisition of list of operations for service group 293  
acquisition of list of operations for service template 140  
acquisition of list of property definitions 262  
acquisition of list of property groups 287  
acquisition of list of property values 275  
acquisition of list of schedules 190  
acquisition of list of service groups 290  
acquisition of list of service templates 134  
acquisition of list of services 155  
acquisition of list of steps 245  
acquisition of list of tag groups 295  
acquisition of list of tags 296  
acquisition of list of task operations 216  
acquisition of list of tasks 208  
acquisition of lists of property definitions and property values 272  
acquisition of property definition information 269  
acquisition of property value 282  
acquisition of schedule information 193  
acquisition of service information 157  
acquisition of task log 247  
acquisition of URL for details of service 184  
acquisition of URL for displaying details of service template 149  
acquisition of user information 300  
acquisition of version information 301  
API 90  
API description format 133  
API functions for information management 300  
API usage example 303  
archiving task 243

## B

backsystem (backing up JP1/AO system) 75  
batch update of property values 278

## C

cancellation of schedule 198  
change of status of service to disabled 182  
change of status of service to maintenance 179  
change of status of service to release 176  
change of version of service template used by service 187  
command description format 19

- commands 13
  - configuration-related commands 20
  - description format 19
  - list 14
  - maintenance-related commands 75
  - notes on using 16
  - operation-related commands 33
  - valid characters for arguments 18
- communication protocol 96
- configuration-related commands 20
- creation of service based on service template 152

## D

- deleteremoteconnection (deleting connection destination definition) 33
- deleteservicetemplate (deleting service template) 35
- deletion of history records (with conditions specified) 255
- deletion of history records (with ID specified) 259
- deletion of service 162
- deletion of service template 139
- deletion of task 215
- domain name and resource that can be managed by API 100
- domain object format 105

## E

- editing service 159
- encryptpassword (creating password file) 20
- error information 131
- execution of service 169
- export of service template 147

## F

- forced stoppage of task 224

## H

- hcnds64checkauth (verifying connection with external authentication server) 21
- hcnds64chgurl (updating URL information) 37
- hcnds64dbrepair (re-creating database) 76
- hcnds64dbsrv (starting and stopping databases) 78
- hcnds64dbtrans (backing up and restoring databases) 79
- hcnds64fwcancel (adding exception to Windows Firewall exceptions list) 24
- hcnds64getlogs (collecting log information) 82

- hcnds64intg (deleting or checking authentication data) 24
- hcnds64ldapuser (registering and deleting user for LDAP search) 26
- hcnds64srv (starting and stopping JP1/AO, and displaying status of JP1/AO) 38
- hcnds64ssltool (creating private key and self-signed certificate) 28
- hcnds64unlockaccount (unlocking user account) 41

## I

- import of service template 144
- importservicetemplate (importing one or more service templates) 43
- input/output format 97

## L

- list of APIs 91
- list of history-related API functions 253
- listremoteconnections (outputting connection destination definition list) 45
- listservices (outputting service or service template list) 48
- listtasks (outputting task list and detailed task information) 52

## M

- maintenance-related commands 75
- member of resource 107
- member to be returned for API functions that acquire executable operation 129
- member to be returned for API that execute JP1/AO operation 128

## N

- namespace 97

## O

- operation-related commands 33

## P

- pause of schedule 201
- property-related API 262

## Q

- query parameter 100

## R

- re-execution of task [228](#)
- reference information [309](#)
- request format [97](#)
- request header [102](#)
- reset of counter for service [173](#)
- response format [99](#)
- response header [106](#)
- response to task that is in the status Waiting for Response [232](#)
- restoresystem (restoring JP1/AO system) [86](#)
- resume of schedule [205](#)
- retry from failed step [236](#)
- retry from step after failed step [239](#)

## S

- schedule-related API [190](#)
- security and authentication [96](#)
- service group-related API functions [290](#)
- service template-related API function [134](#)
- service-related API [155](#)
- setremoteconnection (adding or updating connection destination definition) [60](#)
- setupcluster (configuring cluster environment) [30](#)
- specification common to API [96](#)
- status code [131](#)
- stoppage of task execution [220](#)
- stoptask (stopping task) [63](#)
- submittask (executing service and re-registering tasks in a batch) [65](#)
- supported method [99](#)

## T

- tag-related API functions [295](#)
- task-related API [208](#)

## U

- update of property value [283](#)
- using HQL standard [104](#)

---

 **Hitachi, Ltd.**

6-6, Marunouchi 1-chome, Chiyoda-ku, Tokyo, 100-8280 Japan

---