Operations Orchestration

Software Version: 10.80

Windows and Linux

Develop

Document Release Date: September 2017

Software Release Date: September 2017

Hewlett Packard Enterprise

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hpe.com/.

This site requires that you register for an HP Passport and to sign in. To register for an HP Passport ID, click **Register** on the HPE Software Support site or click **Create an Account** on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HPE sales representative for details.

Contents

Develop

You can navigate about the guidelines for developing new operations for the Operations Orchestration platform. Operations are used to create new flows that can be executed inside OO flow execution engine.

Introduction

This document describes HPE Operations Orchestration public Application Programming Interfaces (API).

The public API is HTTP-based. All APIs are RESTful and use JavaScript Object Notation (JSON).

REST API Versions

Some previous releases of HPE OO 10 have introduced new REST API versions. When a new version is introduced, previous versions are deprecated, but still supported for a period of time. The current version is v2 and the APIs documented in this guide are updated for it. The version should be specified as a prefix to all REST calls (e.g. /oo/rest/v2/executions).

Note: New HPE OO releases may introduce new optional fields to returned representations that are not considered as an API break.

Therefore, the client should be tolerant to new attributes when deserializing JSONs. For example, when performing a GET operation on /test, the following is returned:

{

"msg" : "hello world"

}

We may add a new attribute in the next version and then the GET on /test will return the following, in this case the client should not break:

{

"msg" : "hello world",

"msg2" : "It’s a beautiful day! "

}

Deprecated API's

HPE is committed to support deprecated APIs until the next major release.

Basic Concepts

See the *Concepts Guide* for more information on the basic concepts of OO.

RESTful APIs

All REST APIs have a prefix of /oo/rest/v2/. For example, POST /oo/rest/v2/executions.

Request Headers

The content-type and accept headers are usually added for every request.

The content-type represents the MIME (RFC2045) type of the request body. The content-type is usually application/json unless otherwise stated in a specific API.

The accept header represents the requested format of the response from the Central server. The accept header is also usually application/json unless mentioned differently.

Integration Use Case

This chapter describes a common usage of the HPE OO API and comes to demonstrate its capabilities. Keep in mind that use case described here is only one example on a common use case of HPE OO platform integration. HPE OO APIs allow much more than that.

Use Case Description

The most common use case when integrating with HPE OO is allowing various types of end users to invoke automation using organizational portal or a third party application. For example, to remediate an

incident, doing routine tasks like reset password for a user or creating a DB schema in Dev environment, and so on.

The following implementation is a suggestion and can be adopted at any level you see fit.

Use Case Implementation

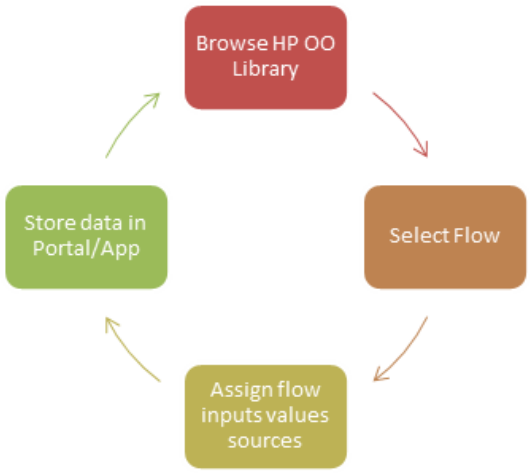
The integration includes two separate processes. These processes are described from the user perspective, but also describe the work to be done by the integration developer.

Portal/App Admin Process - Selecting Flows to Expose to Users

Process description

Before the user of the Organizational Portal/Application will be able invoke flows from it, the Admin needs to determine which flows he would like to expose to the user and for each one of them to determine from where the user is able to invoke and assign data sources for the flow inputs.

The Admin experience is:



Process implementation

This process, if used as described, requires UI development on the portal/application side in order to allow the Admin to browse the library and select a flow.

For example:

- Drop down selection box that lists all the flows in a specific folder in the library (means that the path will need to be decided in advance).
- A folders tree graphical window like the following:

Another option which is less usable for the Admin is only supplying the UI that allows the Admin to manually insert the flow UUID and input parameters value sources.

The following table describes how the implementation of the interactions with HPE OO Central server look like.

Step: Browse HPE OO Library

Admin Action: Browse the content library from the portal/application.

Integrator Actions (interaction with OO): Lists the folders and flows under a given path while the root of the path is the HPE OO Content Library, which is 'Library/'. In this example, if the organizational portal team decides to implement in the UI the full library tree display (as shown in the image above), the integrator code is required to be recursive. That is, a REST call will be implemented for every branch that the end user clicks.

Looking at the example in the image above, the first REST call was to list the top level libraries, then when the user clicked on 'Accelerator Packs' a REST call was submitted to list the levels below it, and so on.

See API: GET /flows/tree/level

Step: Select Flow

Admin Action: Select flows to invoke in order to make them available in the portal/application and also define where.

Integrator Actions (interaction with OO): Get the selected flow details like UUID, Inputs, Description, etc. The details that will be collected depend on what information was decided to display to the Admin in the UI. For invoking the information needed is UUID and inputs information.

See API: GET /flows/{uuid}

Step: Assign flow inputs values sources

Admin Action: Bind value sources to the flow inputs. The sources will most likely be dynamic objects from the application data (like internal variable, called SelectedItemHostname) and not static values.

Integrator Actions (interaction with OO): Provide the capability for this in the portal/application. Note: A validation will need to be implemented to make sure the Admin will provide value source to each of the flow inputs that are marked as Prompt User. Otherwise the flow will pause and will wait for inputs, for example, OO Admin will need to login to Central and enter them.

See API: GET /flows/{uuid}/inputs

Step: Store data in Portal/App

Admin Action: Store all the information in the Portal/Application.

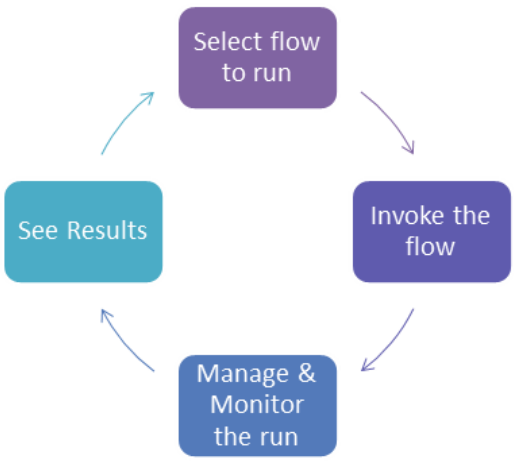
Integrator Actions (interaction with OO): Save the relevant data to the portal/application (in its DB/Forms/Files/etc.)

Note: The flow UUID, inputs and their value source must be kept on the Portal/Application side for the flow invocation.

End User process - Invoking and Monitoring Workflows

Process description

This process occurs in the organizational portal or the third party applications, on the area that is exposed to the end user. The best practice is to have one place that holds the functionality, like an internal service, so the other areas of the application that allow users to trigger flows calls it over and over.



Process implementation

The following table describes how the implementation of the interactions with HPE OO Central server looks like.

The interaction is done through the HPE OO REST API.

See Flow Execution for more details.

Step: Select flow to run.

End User Action: From the portal/application, the end user will select the flow to invoke from a predefined list or just click on a button that the admin made available.

Integrator Actions (interaction with OO): Collect the information to be used later for invoking the flow. This includes UUID of the flow selected and input parameters designated values.

API to use: None

Step: Invoke the flow.

End User Action: The workflow will be invoked while the portal/application will feed it with the needed input values.

Integrator Actions (interaction with OO): Use the REST API to invoke the flow. Use the UUID and the flow input parameters names and values. It is also recommended to use the runName invocation parameters in order to allow better troubleshooting later on. A suggested format for the runName can be:

<InvokingAppName>:<InvokingUserName>:<TargetSystemName>:<ActionName>

API to use: Execute a Flow by UUID POST /executions

Basic Authentication

When user authentication is on, the client must provide their credentials when calling the REST APIs. Central supports preemptive basic authentication.

The client should add a header with the following key/value:

- Key: Authorization
- Value: Basic base64 (username:password)

For example, the authorization value for admin:1234 is: Basic YWRtaW46MTIzNA==

On an unsuccessful authentication attempt, the service returns an HTTP 401 code.

Permissions

If the user does not have the assigned permission to activate an API the following status code appears:

```
403 – Forbidden. The user attempting to execute this command does not have the
needed permission.
```

> This applies to all the API's in the version.

CSRF Protection in OO 10.x

A CSRF (Cross-Site Request Forgery) attack is when a malicious web site, email, blog, instant message, or program causes a user's web browser to perform an unwanted action on a trusted site for which the user is currently authenticated. The impact of a successful cross-site request forgery attack is limited to the capabilities exposed by the vulnerable application.

In a fresh installation of OO, or when you upgrade from an earlier version, CSRF protection is enabled by default.

For more information about CSRF protection, see https://www.owasp.org/index.pHPE/Cross-Site_Request_Forgery_%28CSRF%29_Prevention_Cheat_Sheet#Disclosure_of_Token_in_URL .

Important: If you are working with a REST client that retains the session, continue reading this section. If not, then this section is not relevant for you.

Adding the CSRF Token Header

When using REST APIs, in some situations, you will need to add the CSRF token header (just for POST/DELETE/PUT).

The POST, PUT, and DELETE requests are protected.

If you are working with a REST client that keeps the session you will need to provide the CSRF token for all the **POST/PUT/ DELETE** requests.

1. The CSRF token can be obtained after double-submission of **GET** requests on any OO valid API. This was changed from the initial CSRF implementation which was in 10.50 release.
   For example, you can submit 2 **GET** requests on the API `/oo/rest/version`. Even if a token is

generated from the 1st call, only the token from the 2nd response is a valid one that can be used in further **POST/ PUT / DELETE** calls.
The response headers contain fields such as:

```
X-CSRF-HEADER: X-CSRF-TOKEN
X-CSRF-PARAM: csrf
X-CSRF-TOKEN: 5a81b051-1a73-4b87-85ad-7c672803c65f
```

2. You can then take this token and put it as a Request Parameter in the **POST** call:

```
X-CSRF-TOKEN: 5a81b051-1a73-4b87-85ad-7c672803c65f
```

> To demonstrate how the CSRF token can be obtained correctly, a sample flow is created in the HPE Solutions Content Pack (latest version), under : `Library/Integrations/Hewlett-Packard/Operations Orchestration/10.x/Samples/Launch flow with HTTP Cookies and CSRF token`.

REST APIs

This section includes the RESTful APIs used in HPE Operations Orchestration.

Flow Execution

These APIs enable you to execute flows.

Get Executions Summary (with pagination and filtering)

Request: GET /executions

Description: Returns a paginated list of executions summary, with filtering. The returned objects include the execution summary, detailed objects with data about the execution.

Request parameters:

Attribute	Type	Description	Required
pageNum	Integer	The number of the returned page.	No

		Default value: 1	
pageSize	Integer	The size of the returned page. Default value: 200	No
flowPath	String	Return runs that their flow path contains this string.	No
status	Array of Predefined Values	Return runs with the following statues. Possible values: RUNNING, COMPLETED, COMPLETED_ RESOLVED, COMPLETED_DIAGNOSED, COMPLETED_ERROR, COMPLETED_NO_ACTION_TAKEN, COMPLETED_ CUSTOM, SYSTEM_FAILURE, PAUSED, PAUSED_ USER_PAUSED, PAUSED_INPUT_REQUIRED, PAUSED_ INPUT_REQUIRED_MANUAL_OP, PAUSED_DISPLAY, PAUSED_GATED_ TRANSITION, PAUSED_HAND_OFF, PAUSED_INTERRUPT, PAUSED_NO_ WORKERS_IN_GROUP, PAUSED_ BRANCH_PAUSED, CANCELED	No
owner	String	Return runs when the owner name contains this string.	No
runName	String	Return runs with a run name that contains this string.	No
runId	String	Return runs when the run id contains this string.	No
flowUuid	String	Return runs where the flow UUID contains this string.	No
startedAfter	Long	Returns runs where the start time is after this time.	No
startedBefore	Long	Returns runs where the start time is before this time.	No

Get Execution Summary

Request: GET /executions/{executionIds}/summary

Description: Retrieves the details of a specific execution.

Example:

GET /executions/3332190961082830376,679861347442169334/summary

Request path variables:

Attribute	Description

executionIds	The ids of the executions

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested execution log was.
403	Forbidden	
404	Not Found	The requested execution log was not found.

Response entity body:

- on success: Returns a JSON object with the following format:

[{

"executionId":"3332190961082830376", "branchId":null, "startTime":1371475041169, "endTime":null,

"status":"PAUSED", "resultStatusType":"RESOLVED", "resultStatusName":"HAHA", "pauseReason":"USER_PAUSED", "owner":"anonymous", "ownerDomain:null", "triggeredBy":"anonymous",

"flowUuid":"a8e8fc10-b584-4d39-921f-987b29c9dd19", "flowPath":null,

"executionName":"mock flow", "triggeringSource:central" "roi":null

},

{

"executionId":"679861347442169334", "branchId":null, "startTime":1371475041169, "endTime":null,

"status":"PAUSED", "resultStatusType":"RESOLVED", "resultStatusName":"HAHA", "pauseReason":"USER_PAUSED", "owner":"anonymous", "ownerDomain:null", "triggeredBy":"anonymous",

"flowUuid":"a8e8fc10-b584-4d39-921f-987b29c9dd19",

"flowPath":null, "executionName":"mock flow", "triggeringSource:central" "roi":null

}

]

See returned items in the Get Execution API for more information.

Get Execution Step Count

Request: GET /executions/{executionId}/steps/count

Description: Returns the total number of executed steps for the given execution, including finished steps and currently executing (or paused) steps. If a step has been executed more than once (e.g., in a loop), this will be reflected in the result. The count includes virtual steps such as lanes.

This command is useful when retrieving steps with pagination. It allows one to compute the total number of pages, or the page number in which a given step could be found.

Request path variables:

Attribute	Description
executionId	Execution ID whose steps to count.

Request parameters:

Attribute	Type	Description	Required	Default
includeRerunHistory	Boolean	Only relevant to reruns. If true, the result will include the steps of the original run (if not purged yet)	No	False
upToPath	String	If provided, the result will only count steps positioned before this path (exclusive). The given value is a step path in the execution tree: The paths of the steps at the top level of the flow are 0.0 for the 1st step, 0.1 for the 2nd and so on. If the 2nd step (0.1) is a subflow, then the paths of the steps in that subflow are 0.1.0, 0.1.1 and so on.	No	N/A

Example:

GET /executions/100300001/steps/count?upToPath=0.7.0.13

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Successful. If there are no results, the result will be empty, but this is still OK.
400	Bad Request	If any of the arguments are invalid.
404	Not Found	If the execution cannot be viewed or does not exist.

Get Execution Steps (with Pagination and Filtering)

Request: GET /executions/{executionId}/steps

Description: Returns a paginated list of executed steps, with optional filtering. If a step has been executed more than once, for example, loop, this will be reflected in the result. The result includes virtual steps such as lanes.

The returned objects are step logs – detailed objects with all available data about the steps. Step logs are created as soon as the step begins to execute, and are updated with more data when the step finishes.

Request path variables:

Attribute	Description
executionId	Execution ID whose steps to retrieve.

Request parameters – pagination (optional):

Attribute	Type	Description	Default	Required
pageNum	Long	The wanted page number (1 or greater).	1	No
pageSize	Long	The size of each page – can be 1 to 1000.	50	No
order	Predefined Value	The order of the returned steps – asc for ascending or desc for descending. The order is by step path.	asc	No

Request parameters – filtering criteria:

General notes about filtering:

- All filtering parameters are optional
- Only steps that satisfy all of the given criteria will be returned
- Filtering is performed before the pagination. In other words, when filtering, the pagination will give you pages of search results.
- Filtering by text is case insensitive
- Range conditions (such as roiFrom and roiUpTo) are exclusive – see the example below for details.

Attribute	Type	Description
includeRerunHistory	Boolean	Only relevant to reruns. If true, the result will include the steps of the original run (if not purged yet)
path	String	Get the step with this exact path. See Get Execution Step Count.
pathFrom	String	Get steps whose paths are greater than this path. Greater refers to a collapsible tree representation of the execution, the greater path would show up lower in the tree. For example, 0.10.0 is greater than both 0.10 (its parent) and0.9.99999.
pathUpTo	String	Get steps whose path is less than this path (also see pathFrom).
nameContains	String	Get steps whose names contain this substring.
types	Predefined Value	Get steps of the given types (comma-separated list, no spaces). The types are: operation, subflow, return_step, other. other represents special steps like Multi Instance, as well as lanes.
startTime	Long	Get steps that started at this exact timestamp. Timestamps are "Unix time" numbers with millisecond resolution (the number of milliseconds elapsed since 00:00:00 1 January 1970 UTC).
startTimeFrom	Long	Get steps that started after this timestamp.
startTimeUpTo	Long	Get steps that started before this timestamp.
endTime	Long	Get steps that ended at this exact timestamp.
endTimeFrom	Long	Get steps that ended after this timestamp.

Attribute	Type	Description
endTimeUpTo	Long	Get steps that ended before this timestamp.
durationSec	Long	Gets steps with this exact execution time (seconds).
durationSecFrom	Long	Get steps that took longer to execute than this value (seconds).
durationSecUpTo	Long	Get steps that took less time to execute than this value (seconds).
inputsContain	String	Get steps where any of the inputs contain this sub-string, in either the input name or its value. It is also possible to make a search in the form name=value. Note: Values over 4,000 bytes cannot be searched.
rawResultsContain	String	Get steps where any of the raw results contain this sub-string, in either the result name or its value. It is also possible to make a search in the form name=value. Note: Values over 4,000 bytes cannot be searched.
primaryResultContains	String	Get steps where the primary result contains this sub- string. Note: Values over 4,000 bytes cannot be searched.
stepResultsContain	String	Get steps where any of the step-defined results contain this sub-string, in either the result name or its value. It is also possible to make a search in the form name=value. Note: Values over 4,000 bytes cannot be searched.
responseTypes	Predefined Value	Get steps having one of the given response types (comma-separated list, no spaces). The types are: resolved, error, diagnosed, no_action_taken, exception. exception means the step's execution could not be completed.
transitionContains	String	Get steps whose outgoing transition messages contain this sub-string. The transition message is either the transition's description, or the transition's name if no description has been defined. Note: Transition descriptions over 4,000 bytes (for all locales combined) cannot be searched.
roi	Double	Get steps whose outgoing transitions have this exact ROI value.

Attribute	Type	Description
roiFrom	Double	Get steps whose outgoing transitions have ROI values greater than this.
roiUpTo	Double	Get steps whose outgoing transitions have ROI values less than this.
currentFlowContains	String	Get steps that belong to flows whose names contain this sub-string.
currentFlowIdContains	String	Get steps that belong to the flow with the given ID (or a sub-string in it).
stepIdContains	String	Get steps with the given static step ID (or a sub-string in it). This ID is hard-coded in the parent flow definition. Note that the same step ID may appear in multiple steps of the execution, for example, when the flow employs a loop. To get a specific step in the execution tree, use the path parameter.
invokedIdsContain	String	Get steps that invoke (execute) the operation or subflow with the given ID (or a sub-string in it). Note: for steps that run soft copies of operations, the invoked IDs include both the soft copy and its parent operation.
userContains	String	Get steps that executed while the given Central user name (or a sub-string in it) was the execution owner.
workerIdContains	String	Get operation-type steps that were executed by the given worker UUID (or a sub-string in it).
workerGroupContains	String	Get operation-type steps that were executed by a worker in the given worker group name (or a sub-string in it).

Example:

GET /executions/100300001/steps?pageNum=3&pageSize=20&order=desc

&pathFrom=0.7.2&pathUpTo=0.9&path=0.9

&nameContains=ping

&types=operation,subflow

&startTime=1391359774000&startTimeFrom=1391359774000&startTimeUpTo=1391359780000

&durationSecFrom=300&durationSecUpTo=400&durationSec=400

&inputsContain=localhost

&resultsContain=ping+completed

&responseTypes=resolved,diagnosed

&transitionContains=success

&roi=7.5&roiFrom=7.5&roiUpTo=30

&currentFlowContains=My+Subflow

&userContains=jon

&workerIdContains=5c2002da

This will search execution 100300001 for steps satisfying all of the criteria, and return the 3rd page out of the result set (with 20 steps per result page). The steps will be searched in descending path order, starting with the step that has the greatest path (normally the main flow's return step).

The combination of these three conditions: roi=7.5&roiFrom=7.5&roiUpTo=30, will search for: 7.5 ≤roi < 30.

Following are the possible search ranges:

Use paramaters…	To search for…
roi=7.5	roi = 7.5
roiFrom=7.5	roi > 7.5
roi=7.5&roiFrom=7.5	roi ≥ 7.5
roiFrom=7.5&roiUpTo=30	7.5 < roi < 30
roi=7.5&roiFrom=7.5&roiUpTo=30	7.5 ≤ roi < 30
roiUpTo=7.5&roiFrom=30	roi < 7.5 or roi > 30
roiUpTo=7.5&roiFrom=7.5	roi ≠ 7.5

Get a Single Execution Step

Request: GET /executions/{executionId}/steps/{stepPath}

Description: Returns data about a single step of an execution. For more details, see Get Execution Steps (with Filtering and Pagination).

Request path variables:

Attribute	Description
executionId	Execution ID whose step to retrieve.
stepPath	The step's path in the execution tree. Step paths work as follows: The paths of the steps at the top level of the execution are 0.0 for the 1st step, 0.1 for the 2nd and so on. If the 2nd step (0.1) is a subflow, then the paths of the steps in that subflow will be 0.1.0, 0.1.1 and so on. Note: Virtual steps such as lanes also have a path.

Example:

GET /executions/100300001/steps/0.7.0.13

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Successful.
400	Bad Request	Any of the arguments are invalid.
404	Not Found	The requested step wasn't found in the given execution, or the execution is not viewable

Get Execution Steps as a CSV File

Request: GET /executions/{executionId}/steps?mediaType=csv

Description: Returns all executed steps for the given execution, as a CSV file.

Request path variables:

Attribute	Description
executionId	Execution ID whose steps to retrieve.

Request parameters:

The mediaType parameter must be set to csv. Otherwise, the request will be treated as Get Execution Steps (with Filtering and Pagination).

Additional parameters:

Attribute	Type	Description	Required	Default
includeRerunHistory	Boolean	Only relevant to reruns. If true, the result will include the steps of the original run (if not purged yet)	No	false

Examples:

GET /executions/100300001/steps?mediaType=csv

GET /executions/100300001/steps?mediaType=csv&includeRerunHistory=true

Code	Meaning	Returned When
200	Successful (OK)	Successful. If there are no results, the result will be empty, but this is still OK.
404	Not Found	If the execution cannot be viewed or does not exist.
400	Bad Request	If any of the arguments are invalid.

Get Execution Steps as a Text File

Request: GET /executions/{executionId}/steps?mediaType=text

Description: Given an execution ID, returns the text representation of all executed steps in a format similar to the HPE OO 9 version.

Request path variables:

Attribute	Description
executionId	Execution ID whose steps to retrieve.

Request parameters:

The mediaType parameter must be set to text. Otherwise, the request will be treated as Get Execution Steps (with Filtering and Pagination).

Additional parameters:

Attribute	Description	Required	Default
includeRerunHistory	Only relevant to reruns. If true, the result will include steps of the original run(if not purged yet).	No	false

Examples:

GET /executions/186400079/steps?mediaType=text

GET /executions/186400079/steps?mediaType=text&includeRerunHistory=true

Response status codes:

Code	Meaning	Returned When
200	OK	Successful
404	Not Found	The specified execution does not exist or is not viewable by the current user.
400	Bad request	If any of the arguments are invalid.

Get Rerunnable Steps

Request: GET /executions/{executionId}/rerunnable-steps

Description: Given an execution ID, returns the set of steps from which the execution can be rerun.

Example:

GET /executions/186400079/rerunnable-steps

Response status codes:

Code	Meaning	Returned When
200	OK	Successful
404	Not Found	The specified execution does not exist or is not viewable by the current user.

Response entity body:

An array of execution step paths (as described in Get Execution Step Count), from which the execution can be rerun. If the execution is not currently in a rerunnable state, or has no rerunnable steps, the result will be an empty array.

Example:

["0.2", "0.3.0.5", "0.7.2"]

Rerun from step

Request: POST /executions/rerun

Description: Given an execution ID and a rerunnable step path in that execution, reruns from that step.

Example:

POST /executions/rerun

Body:

Attribute	Type	Description	Required?	Default
executionId	String	Execution to rerun	Yes	
stepPath	String	Step path to rerun from	Yes	
runName	String	Execution run name	No	Flow name
logLevel	String	Execution log level. The logLevel attribute receives one of these values: STANDARD, EXTENDED	No	Flow log level

Example:

{

"executionId":"101023434",

"stepPath":"0.0",

"logLevel":"STANDARD",

"runName": "rerun_on_SYSTEM_PROPERTY"

}

Response status codes:

Code	Meaning	Returned When
200	OK	Successful
404	Not Found	The specified execution does not exist or is not viewable by the current user.
403	Forbidden	1. Rerun functionality is disabled 2. Rerunning is not supported for this run's current state or type. 3. No permissions

Response entity body:

New execution ID

Example:

101023436

Delete Rerun data

Request: DELETE /executions/rerun

Description: Given the ended before date and max amount of execution – deletes the rerun data from DB.

Request parameters:

Attribute	Type	Description	Required	Default
endedBefore	Long	Time to use as upper limit for purging	Yes	
maxAmount	Long	Max number of executions to purge	No	100

Example:

DELETE /executions/rerun?endedBefore=1445421975041&maxAmount=1000

Response status codes:

Code	Meaning	Returned When
200	OK	Successful

400	Bad Request	The parameters are not correct
403	Forbidden	No permissions

Response entity body:

Number of executions that were deleted

Example:

1337

Execution Pauses

Request: GET /executions/{executionId}/pauses

Description: Retrieves current pauses for the given execution id.

Request path variables:

Attribute	Description
executionId	The ID of the execution which the client wishes to retrieve its pauses.

Example:

GET /executions/100001/pauses

Response status codes:

Code	Meaning	Returned When
200	OK	All requested pauses were returned.
404	Not Found	The provided execution ID doesn't exist.

Response entity body:

An array which contains all the current pauses of the requested execution. Each element in the array represents a pause of a lane in the execution tree.

The returned array is not ordered and is empty if no pauses exist for the given execution id. There are five possible reasons for pauses, which can be differentiated by inspecting the pauseReason attribute.

Note: There are different return attributes between the different types:

Reason 1: Input Required

Attribute	Type	Description	Comments
pauseReason	Predefined Value	The value INPUT_REQUIRED	
pauseId	Long	An ID for the returned pause.	
executionId	String	The execution ID	
branchId	String	The ID of the branch were the pause has occurred.	null ID represents the main branch.
stepId	String	The UUID of the step in which the pause has occurred.	
stepName	String	The name of the step in the flow.	
requiredInputs	FlowInput	See Get Flow Inputs	

Reason 4: Hand Off

Attribute	Type	Description	Comments
pauseReason	Predefined Value	The value HAND_OFF	
pauseId	Long	An ID for the returned pause.	
executionId	String	The execution ID.	
branchId	String	The ID of the branch were the pause has occurred.	A null ID represents the main branch.
stepId	String	The UUID of the step in which the pause has occurred.	
stepName	String	The name of the step in the flow.	

Example:

[

{

"pauseId":101100009, "executionId":"100100293", "branchId":null,

"stepId":"70eaf376-72ca-4440-9f60-a743fcfa56b2", "stepName":"UUID Generator",

"pauseReason":"HAND_OFF"

}

]

Reason 5: No Workers In Group

Attribute	Type	Description	Comments
pauseReason	Predefined Value	The value NO_WORKERS_IN_GROUP	
pauseId	Long	An ID for the returned pause.	
executionId	String	The execution ID	
branchId	String	The ID of the branch were the pause has occurred.	A null ID represents the main branch.
stepId	String	The UUID of the step in which the pause has occurred.	Would be null, please ignore
stepName	String	The name of the step in the flow.	Would be null, please ignore
groupName	String	The unavailable group.	

Example:

[

{

"pauseId":101100012, "executionId":"100100341", "branchId":null, "stepId":null, "stepName":null,

"pauseReason":"NO_WORKERS_IN_GROUP", "groupName":"RAS_Operator_Path"

}

]

Get Execution Log

Request: GET /executions/{executionId}/execution-log

Description: This API retrieves the extended summary of a specific execution. It is an extension of the Execution Summary API and holds additional information, such as the inputs and outputs of that execution.

Request path variables:

Attribute	Description
executionId	The id of the execution

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested execution log was.
403	Forbidden	
404	Not Found	The requested execution log was not found or not viewable.

Response entity body:

- on success: Returns a JSON object with the following format:

{

"executionSummary":{ "executionId":"348246628680024354", "branchId":null, "startTime":1371366300297, "endTime":null,

"status":"PAUSED", "resultStatusType":"RESOLVED", "resultStatusName":"HAHA", "pauseReason":"USER_PAUSED", "owner":"anonymous", "ownerDomain:null", "triggeredBy":"anonymous",

"flowUuid":"a8e8fc10-b584-4d39-921f-987b29c9dd19", "flowPath":null,

"executionName":"mock flow", "triggeringSource:central"

"roi":null

},

"executionLogLevel": "STANDARD",

Change the Status of an Execution

Request: PUT /executions/{executionIds}/status

Description: Update existing executions statuses. Each request has a single action and data but can be applied to multiple executions.

Note:

Changing the run status is allowed when both of these conditions are true:

- The user has the Run permission (entitlement) for the executed flow.
- The run is currently assigned to the user, or alternatively, the user has the Manage Others' Runs permission.

Request path variables:

Attribute	Description	Type
executionIds	The IDs of the executions which the user wants to update the statuses. Separated by a comma.	String

Request entity body:

The execution status can be changed to one of the following states: CANCEL, PAUSE, REASSIGN, or RESUME. The desired status should be set in the action attribute.

Examples:

To cancel executions:

Attribute	Type	Description	Required	Default
action	Predefined Value	The value CANCEL	Yes	N/A

{

"action":"CANCEL"

}

To pause executions:

Attribute	Type	Description	Required	Default
action	Predefined Value	The value PAUSE	Yes	N/A

{

"action":"PAUSE"

}

For reassigning executions to another user:

Attribute	Type	Description	Required	Default
action	Predefined Value	The value REASSIGN	Yes	N/A

data	Key value	Contains the key userName and the reassigned user as value.	Yes	

{

"action":"REASSIGN",

"data":{

"userName":"John"

}

Execute a Flow by UUID

Request: POST /executions

Description: Executes a flow specified by UUID.

Request entity body:

Attribute	Type	Description	Required?	Default
flowUuid	String	Flow ID	Yes	
runName	String	Execution run name	No	Flow name
logLevel	String	Execution log level. The logLevel attribute receives one of these values: STANDARD, EXTENDED	No	Flow log level
inputs	String	Execution bound inputs	No	
inputPromptUseBlank	Boolean	If true, disables all prompts for input values	No	False

The body of this request must include a JSON object with the following format:

{

"flowUuid":"8d52dfc3-1de5-48d4-9c2a-887718de4696", "runName": "run1",

"logLevel": "STANDARD", "inputs":

{

"input1":"value for input1",

.

.

.

"inputn":"value for inputn"

}

}

- There is an option to change the group alias mapping during a specific flow execution, by specifying an input called OO_ALIAS_GROUP_MAPPING.

For example to map the group alias newAlias to the worker group group name provide the following input:

"inputs":

{

"OO_ALIAS_GROUP_MAPPING":"{\"newAlias\":\"group name\"}"

.

.

.

}

inputs and runName, are optional and can be omitted.

Note: If the flows was defined with inputs which are either set as prompt user or/and are required, the flow will pause.

LogLevel is also optional – if not specified, the default level for the given flow will be used.

The logLevel attribute receives one of these values: STANDARD, EXTENDED.

Flow Execution Status

The following are the possible values of the status attribute, which appears in the APIs:

Status	Description

RUNNING	The flow execution is in progress.
COMPLETED	The flow has finished. In order to understand the result (for example, success or failure) use the resultStatusType attribute.
SYSTEM_ FAILURE	The execution failed due to an unexpected error in the system.
PAUSED	The flow execution paused. For pause reasons, see the Execution Pauses API.
PENDING_ PAUSE	A flow execution pause request was submitted, and the system is waiting for an action to complete in order to enter the pause state.
CANCELED	The flow execution was canceled by the user.
PENDING_ CANCEL	A flow execution cancel request was submitted, and the system is waiting for an action to complete in order to cancel the execution.

Flow Input

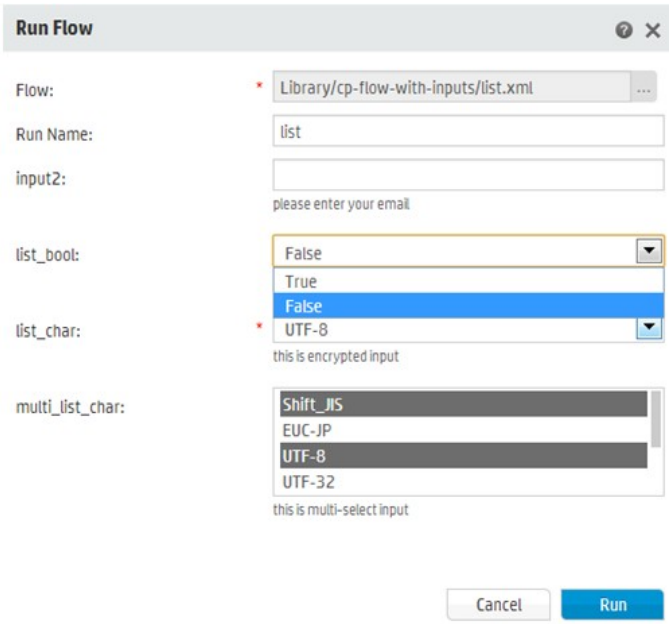
Defines for the client how an input should be presented to the end user.

Attribute	Type	Description	Comments
uuid	String	The UUID of this input.	null is possible
name	String	A unique name of this input.	
valueDelimiter	String	The expected delimiter in the value, in case this is a multiple value input.	null is possible
description	String	A localized description of this input, this provides more information to the user.	
encrypted	Boolean	Indicates whether this is a classified input. It is advised to mask the user input in the presentation.	
multiValue	Boolean	Indicates whether multiple values are expected. They will be delimited by the valueDelimiter attribute.	
mandatory	Boolean	Set if the user must provide this input. In this case the attribute is true and the user does not provide the required input, the operation which requests this input will fail.	
sources	Array	Suggested input values for the user.	null is possible

type	Predefined Value	String: Indicates that a free text input is expected. SelectionList: User should choose value/values from the supplied sources.	
validationId	String	For future use.	Would be null. Please ignore this attribute.
defaultValue	String	A default value for this input. This is a hint for the UI and could be used by the user.	null is possible

Example from Central UI:

input2, list_bool, list_char, multi_list_char are flow inputs.



Example:

[

{

"uuid":"8e1b1288-3f1a-45ef-b23b-cbdf21bb607b", "name":"input2",

"valueDelimiter":",",
"description":"please enter your email", "encrypted":false,

"multiValue":false, "mandatory":false, "sources":null, "type":"String", "validationId":null,
"defaultValue":null

},

{

"uuid":"fdd88ec2-b76f-4aec-a5af-509549bd41fb", "name":"list_bool",

"valueDelimiter":",", "description":"choose yes or no", "encrypted":false, "multiValue":false, "mandatory":false,

"sources":["True",

"False"

],

"type":"SelectionList", "validationId":null, "defaultValue":null

},

{

"uuid":"3406b528-a856-49d1-82b3-516b7c8243c5", "name":"list_char",

"valueDelimiter":",",

"description":"this is encrypted input", "encrypted":false,

"multiValue":false, "mandatory":true, "sources":[

"Shift_JIS", "EUC-JP", "UTF-8",

"UTF-32",

"ISO-2022-JP", "UTF-16", "Windows-31J"

], "type":"SelectionList", "validationId":null, "defaultValue":null

},

{

"uuid":"c4bcf870-a7f9-4160-8be0-eea2fc4978d4", "name":"multi_list_char",

"valueDelimiter":",",

"description":"this is multi-select input", "encrypted":false,

"multiValue":true, "mandatory":false, "sources":[

"Shift_JIS", "EUC-JP", "UTF-8",

"UTF-32",

"ISO-2022-JP", "UTF-16", "Windows-31J"

], "type":"SelectionList", "validationId":null, "defaultValue":null

}

]

Purge Debugger Execution Events

Request: DELETE /debugger-events

Description: Purges debugger events of remote executions.

This command is useful for keeping the database clean from debugger events, which become useless after executions have completed and also affect the size and memory the database occupies.

Request parameters:

Attribute	Type	Description	Required	Default
endedBefore	Long	Time to use as upper limit for purging	Yes	
maxAmount	Long	Max number of executions to purge	No	100

Example:

DELETE /debugger-event?endedBefore=1411998175833

DELETE /debugger-event?endedBefore=1411998175833&maxAmount=1000

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Successful. If there are no results, the result will be empty (which is OK).
400	BadRequest	If any of the arguments are invalid.
403	Forbidden	The user attempting to execute this command does not have the Manage Data Cleanup permission.

Response entity body:

- on success: returns the number of purged executions.

Purge Execution Step Data, Inputs and Outputs

Request: DELETE /executions

Description: Purges execution data for runs matching the given criteria. It is possible to purge all execution data (including the execution summaries), or only specific items (flow inputs, flow outputs and/or steps).

Request parameters:

Attribute	Type	Description	Required	Default value
endedBefore	Long	Time to use as upper limit for purging	Yes	
maxAmount	Integer	Max number of executions to purge	No	100
purgeItems	Predefined Value	List of items that should be purged. Valid values: flowInputs, flowOutputs, steps. If not specified, will purge all data for executions (including execution summaries).	No	
statuses	Predefined Value	List of execution statuses according to which the relevant execution will be purged. Valid values: COMPLETED, COMPLETED_RESOLVED, COMPLETED_DIAGNOSED,COMPLETED_ERROR, COMPLETED_NO_ACTION_TAKEN, COMPLETED_CUSTOM SYSTEM_FAILURE, CANCELED. If not specified, will not filter according to status.	No	
flowUuids	String	Purge data for executions of the given flow UUIDs. If not specified, will not filter according to flow UUID.	No	

Examples:

- DELETE /executions?endedBefore=1445421975041&flowUuids=2da44b9a-2161-47c5-94a9-864920bedef8
  Purges all run data (execution summaries, steps, flow inputs and outputs) for runs of the flow with UUID 2da44b9a-2161-47c5-94a9-864920bedef8 which ended before 2015-10-21 10:06:15 UTC. Up to 100 runs will be purged.

- DELETE /executions?endedBefore=1445421975041&statuses=SYSTEM_FAILURE,CANCELED &flowUuids=2da44b9a-2161-47c5-94a9-864920bedef8
  Purges all run data (execution summaries, steps, flow inputs and outputs) for runs with status SYSTEM_FAILURE or CANCELED which ended before 2015-10-21 10:06:15 UTC. Up to 100 runs will be purged.
- DELETE /executions?endedBefore=1412150176345&maxAmount=200&purgeItems=steps
  Purges step data only, for runs which ended before 2014-10-01 07:56:16 UTC. Up to 200 runs will be purged.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Successful. The requested contents were deleted.
400	Bad Request	If any of the parameters are invalid.
403	Forbidden	The user attempting to execute this command does not have the ManageData Cleanup permission.

Response entity body:

•on success: Returns the amount of executions whose summaries/steps/inputs/outputs were purged.

Delete StepLog data

Request: DELETE /steps-log

Description: Purge step data according to time and amount of executions to purge.

Example: DELETE /steps-log?endedBefore=1412150176345&maxAmount=200

Request parameters:

Attribute	Type	Description	Default value	Required?
endedBefore	Long	The time in milliseconds to start purging	30 days back (in milliseconds)	No
maxAmount	int	The max amount of executions on which step's data is purged	100	No

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The step data was deleted successfully.
400	Bad Request	Arguments are invalid
403	Forbidden	The user does not have the permission Manage Cleanup Data

Response entity body:

- on success: Returns a JSON string of the number of executions that their step data has been purged.

Flow Library

APIs relating to the Flow Library

Get Flows Library

Request: GET /flows/library

Description: Retrieves all flows deployed in the system. This returns an ordered list, and contains all the folders and flows in the flow library, ordered according to the tree structure.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested library was returned.

Response entity body:

A List of elements which contains all of the flows that exist in the system.

Attribute	Type	Description	Comments
id	String	The id of node.	For a non-leaf (folder) node, the id is the same as the path. For a leaf node (deployed entity), the id is the entity's id.
name	String	The name of the node.	Folder name or flow name.
parentId	String	The id of the parent node.	Null if this is a root node.

leaf	Boolean	Whether the node is a leaf or not.	leaf is a flow, not leaf = folder
path	String	The path of the node.	
runnable	Boolean	Whether the node is a runnable or not.	
childrenIds	Array of Strings	List of the the node's children id	

Read Next Level of Library Tree

Request: GET /flows/tree/level

Description: Returns a flat list of all tree Items under the path (lazy loading).

Request parameters:

Attribute	Description	Required
path	Return the tree items under this path. Default value is root.	No

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested items were found

Response entity body:

- on success: Returns a JSON object with the following format:

[

{

"id":"library/Accelerator Packs", "name":"Accelerator Packs", "leaf":false, "path":"Library/Accelerator Packs", "runnable":false,

"children":null

},

{

"id":"library/How Do I flows", "name":"How Do I flows",
"leaf":false,

"path":"Library/How Do I flows", "runnable":false,

"children":null

}

]

Get Flow Details

Request: GET /flows/{uuid}

Description: Returns flow properties by the uuid.

Request path variables:

Attribute	Description
uuid	The flow uuid

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested flow was found.
404	Not Found	The requested flow was not found or the user does not have permission to view it or the uuid was empty

Response entity body:

- on success: Returns a JSON object with the following format:

{

"id":"1fe1be31-2c78-40dd-8326-b8ca527e5587",
"name":"Recently Run",

"path":"Library/Utility Operations/Date and Time/Recently Run.xml", "description":"flow desciption",

"cpName":"HPEOO-oo-base", "version":"version111"

"logLevelInfo":{

"logLevel":"EXTENDED",

"logLevelSource":"SYSTEM"

}

}

Get Flow Inputs

Request: GET /flows/{uuid}/inputs

Description: Retrieves a list of flow's inputs by its UUID.

Request path variables:

Attribute	Description
uuid	The flow uuid

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested flow's inputs were found
404	Not Found	The requested flow was not found or the user does not have permission to view it or the uuid was empty

Response entity body:

- on success: Returns a JSON object with the following format:

[

{

"uuid":"c4454566-6bb5-4be9-9824-2a08945f1574", "name":"message",

"valueDelimiter":",",
"description":"",
"encrypted":false,
"multiValue":false,
"mandatory":true,
"sources":null,
"type":"String",
"validationId":null,
"defaultValue":null

},

{

"uuid":"cdac00b3-f550-4cd5-a3eb-f15d2f80fd78", "name":"title",

"valueDelimiter":",", "description":"", "encrypted":false, "multiValue":false, "mandatory":false, "sources":null,

"type":"String", "validationId":null, "defaultValue":"Status message"

}

]

Get All Flow Inputs

Request: GET /flows/inputs

Description: Gets a list of all deployed flows with all of their inputs.

Example:

GET /flows/inputs

Response status codes:

Code	Meaning	Returned When
200	OK	Successful

Response entity body:

An array of flows where each flow includes the flow UUID, flow name and an arrary of its inputs.

Example:

[

 {

  "flowUuid":"3bb3f887-f4f1-4c4b-a7a3-7e7f4c403283",

  "flowName":"Check Process by PID",

  "flowInputs":[

   {

    "flowUuid":"3bb3f887-f4f1-4c4b-a7a3-7e7f4c403283",

```
    "flowName":"Check Process by PID",
    "inputUuid":"d4237daa-a415-4647-a270-7e0ce14edce2",
    "inputName":"host",
    "inputValueDelimiter":",",
    "inputEncrypted":false,
    "inputMultiValue":false,
    "inputMandatory":true,
    "inputType":"String",
    "inputValidationId":null,
    "inputDefaultValue":null,
    "sources":null
   },
   {
    "flowUuid":"3bb3f887-f4f1-4c4b-a7a3-7e7f4c403283",
    "flowName":"Check Process by PID",
    "inputUuid":"d2a0860a-696d-4696-ba18-65f6b55f747c",
    "inputName":"username",
    "inputValueDelimiter":",",
    "inputEncrypted":false,
    "inputMultiValue":false,
    "inputMandatory":false,
    "inputType":"String",
    "inputValidationId":null,
    "inputDefaultValue":null,
    "sources":null
   }
  ]
 }
```

]

Get Flow Outputs

Request: GET /flows/{uuid}/outputs

Description: Retrieves a list of all the outputs defined for the flow with the requested uuid. The list of outputs retrieved will contain only outputs defined in the flow properties.

Request path variables:

Attribute	Description
uuid	The requested flow's uuid.

Example:

GET /flows/22de91c4-f651-42fd-a404-1bf0ca921f36/outputs

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested flow's inputs were found.
404	Not Found	The requested flow was not found or the user does not have permission to view it or the uuid was empty.

Response entity body:

An array which contains all of the outputs that are defined for the flow. Each element in the array represents an output of the flow.

The returned array is not ordered.

Attribute	Type	Description	Comments
name	String	The name of the output.	

Example:

[

{ "name":"Result"

},

{ "name":"output2"

},

{ "name":"output1"

}

]

Get Flow Settings

Request: GET /flows/{UUID}/settings

Description: Retrieves execution settings for the specified flow.

Request path variables:

Attribute	Description
UUID	The flow UUID

Example:

GET /flows/8aff6b34-15a2-11e5-b60b-1697f925ec7b/settings

Response status codes:

Code	Meaning	Returned When
200	OK	The flow settings were found
404	Not Found	The specified flow does not exist or the user is not entitled to view it

Response entity body:

Attribute	Type	Description
logLevelInfo	Object	The object contains two strings: logLevel: The run log level (persistence level) of this flow. Possible values: STANDARD, EXTENDED

		logLevelSource: Indicates whether the log level is taken from the system default, or is explicitly set for this flow. Possible values: SYSTEM, FLOW
flowTimeout	Integer or null	Possible values: • null: This flow has no explicit timeout setting. The system default timeout will be used instead (see Get General Setting by Key). • 0: This flow is explicitly set to have no timeout. • Positive number: This flow is explicitly set to have the specified execution timeout, in minutes.

Example:

{

"logLevelInfo":{

"logLevel": "STANDARD",

"logLevelSource": "SYSTEM"

},

"flowTimeout": null

}

Update Flow Settings

Request: PUT /flows/{UUID}/settings

Description: Sets execution settings for the specified flow.

Request path variables:

Attribute	Description
UUID	The flow UUID

Request entity body:

See response entity body if

Attribute	Type	Description
logLevelInfo	Object	This object should contain a single property named logLevel. Possible values for logLevel: • null: Remove the explicit log level for this flow (use the system default log level). • One of the predefined logLevel values (see Get Flow Settings).
flowTimeout	Integer or null	See possible values for flowTimeout in Get Flow Settings.

Example:

PUT /flows/8aff6b34-15a2-11e5-b60b-1697f925ec7b/settings

{

"logLevelInfo":{

"logLevel": "STANDARD"

},

"flowTimeout": 60

}

Response status codes:

Code	Meaning	Returned When
204	Successful (No Content)	The flow settings were updated.
400	Bad Request	The request body is invalid.
403	Forbidden	The user does not have the Manage Content Packs permission.
404	Not Found	The specified flow does not exist or the user is not entitled to view it.

Scheduler

The scheduler API allows you to schedule flow executions. You can specify a schedule to run for a specific occasion. You can also setup recurring schedules for a flow for a repeated task. These APIs enable you to manage schedules, for example create new schedules.

Note: The scheduler supports only the quartz CRON syntax.

Create New Flow-Schedule

Request: POST /schedules

Description: Add a new schedule for a flow execution.

Request entity body:

The body of this request must include a JSON object with the following format:

JSON for a scheduled flow with a CRON triggering expression:

```
{
"flowScheduleName":"Scheduled Flow Created By REST", "flowUuid":"c34de7d6-14cc-4a1c-b25e-85afbb064359", "triggerExpression":"0 10 10 ? * 6",
"startDate": 1314079869000,
"endDate": 1491302669536,
"runLogLevel": "STANDARD",
"inputPromptUseBlank": true,
"timeZone": "Asia/Amman",
"inputs":
    {
     "input1":"value for input1",
      .
          .
          .
          "inputn":"value for inputn"
        }
}
```

JSON for a scheduled flow with a simple triggering expression:

{

"flowScheduleName": "Scheduled Flow Created By REST",
"flowUuid":"c34de7d6-14cc-4a1c-b25e-85afbb064359",
"triggerExpression":"*/60000",
"startDate": 1314079869000,
"endDate": 1491302669536,
"numOfOccurrences": 5,
"runLogLevel": "STANDARD",
"timeZone": "Asia/Amman",
"inputPromptUseBlank": false,

"inputs":{

"input1": "value for input1",

.

.

.

"inputn": "value for inputn"

}

}

- If endDate is not set, by default, it receives a value of 0.

- If username is not set, by default, it receives a value of null.
- The trigger expression should be either a valid cron expression or a simple expression according the pattern below.
- If you use the cron expression, you can validate it using an expression validity.
- If you want to use a simple trigger expression (every x minutes) you should use the syntax according to the following example:

*/6000 = run every 60000 milliseconds (every minute)

Note: If you use a cron expression you cannot add the numOfOccurences attribute as it may conflict with the cron expression. In addition, if you use simple triggers and add both end time and number of occurrences, the triggering ends according to the number of occurrences.

Response status codes:

Code	Meaning	Returned When
201	Created	A schedule was created successfully.
400	Bad Request	The user does not have Manage Schedules permission.
403	Forbidden	

Response entity body:

- on success: Returns a JSON object of the created schedule with the following format:

{

"id": "1347298851037",

"flowScheduleName": "Scheduled Flow Created By REST",
"flowUuid": "c34de7d6-14cc-4a1c-b25e-85afbb064359",
"triggerExpression": "*/60000",
"startDate": 1314079869000,
"endDate": 1491302669536",

"username": "DavisJ",

"numOfOccurrences": 5,
"runLogLevel": "STANDARD",
"timeZone": "Asia/Amman",
"nextFireTime": -1,
"prevFireTime": -1,

"enabled": false,
"inputPromptUseBlank": false,

"inputs":{

"input1": "value for input1",
.
.
.

"inputn": "value for inputn"

}

}

Enable or Disable Flow-Schedule

Request: PUT /schedules/{ids}/enabled

Description: Enable or disable existing flow-schedules.

Request path variables:

Attribute	Description
ids	The identifiers of the flow-schedules to enable or disable.

Request entity body:

true or false

Response status codes:

Code	Meaning	Returned When
204	Successful (No Content)	The flow-schedules were updated successfully.
400	Bad Request	
403	Forbidden	The user does not have Manage Schedules permissions.

Request entity body:

The body of this request needs to include a JSON value of either true to enable the schedules or false to disable them.

Delete Flow-Schedule

Request: DELETE /schedules/{ids}

Description: Deletes flow-schedules according to the specified IDs.

Request path variables:

Attribute	Description
ids	The identifiers of the flow-schedules to delete.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	All specified flow-schedules were either deleted successfully, or did not exist to begin with.
400	Bad Request	
403	Forbidden	The user does not have Manage Schedules permissions.

Response entity body:

- on success: Returns a JSON array of the deleted schedule IDs

Get Flow-Schedules

Request: GET /schedules

Description: Returns all existing flow-schedules headers.

Response status codes:

Code	Meaning	Returned When
200	OK	The requested flow-schedules were found.
403	Forbidden	The user does not have View Schedules or Manage Schedules permission.

Request parameters:

Attribute	Type	Description	Is required?	Default value
start	Integer	The page number to return (starting with 1 for the first page of results).	No	1
pageSize	Integer	The number of results in each page.	No	50
direction	Predefined Value	The direction in which to sort the results: ASC for ascending or DESC for descending.	No	ASC
orderBy	Predefined Value	The schedule property by which to sort the results: scheduleName, flowName, nextFireTime, prevFireTime, scheduleState or username.	No	scheduleName
filter	String	String to filter by.	No	No filter

Example:

GET /schedules?start=3&pageSize=20&direction=DESC&orderBy=nextFireTime&filter=my-schedule

Response entity body:

- on success: Returns a JSON array, containing all existing flow-schedules headers, with the following format:

[

{

"id":"123", "enabled":true,

"flowUuid":"78bec456-db6a-4c05-99ad-0675b230bfeb",

"nextFireTime":-1,
"prevFireTime":-1, "flowScheduleName":"schedule 1", "flowName":"flow1", "flowPath":"path0",
"username":"admin" "triggerExpression":"0 10 10 ? * 6"

},

.

.

.

{

"id":"567", "enabled":true,

"flowUuid":"3d32e475g-ab54-fe21-df32-4743346ebebd", "nextFireTime":-1,

"prevFireTime":-1, "flowScheduleName":"schedule n", "flowName":"flow3", "flowPath":"path2",
"username":"admin" "triggerExpression":null

}

]

Get Flow-Schedule Details

Request: GET /schedules/{id}

Description: Returns details about a flow-schedule specified by ID.

Request path variables:

Attribute	Description
id	The identifier of the flow-schedule to retrieve.

Response status codes:

Code	Meaning	Returned When
200	OK	The requested flow-schedule was found.
403	Forbidden	The user does not have View Schedules or Manage Schedules permission.
404	Not Found	The requested flow-schedule was not found.

Response entity body:

- on success: Returns a JSON object of the flow-schedule details with the following format:

{

"id": "1399455773960",

"flowScheduleName": "Scheduled Flow Created By REST",

"flowUuid": "c34de7d6-14cc-4a1c-b25e-85afbb064359",

"triggerExpression": "0 10 10 ? * 6",

"startDate": 1399455780000,

"endDate": 1491302669536,

"numOfOccurrences": null,

"timeZone": "Asia/Amman",

"username": "admin",

"runLogLevel": "STANDARD",

"nextFireTime": 1399619400000,

"prevFireTime": -1,

"enabled": true,

"inputPromptUseBlank": false,

"inputs":{

"input1": "value for input1",
"input2": "value for input2"

}

}

Update Flow-Schedule

Request: PUT /schedules/{id}

Description: Updates an existing flow-schedule. Includes a list of values that can be updated.

Request path variables:

Attribute	Description
id	The identifier of the flow-schedule to update.

Request entity body:

The body of this request needs to include a JSON object with the following format:

{

"flowScheduleName": "Scheduled Flow Created By REST",
"flowUuid": "c34de7d6-14cc-4a1c-b25e-85afbb064359",
"triggerExpression": "0 10 10 ? * 6",
"startDate": 1376072040000,
"endDate": 1377334800000,

"runLogLevel": "STANDARD",
"timeZone": "Asia/Amman",
"inputPromptUseBlank": false,

"inputs":{

"input1": "value for input1",

.

.

.

"inputn": "value for inputn"

}

}

Response status codes:

Code	Meaning	Returned When
200	OK	Returned when update schedule successfully
400	Bad Request	
403	Forbidden	The user does not have Manage Schedules permission.

Response entity body:

- on success: Returns a JSON value: true

Dashboard

The Dashboard workspace reflects the system's ROI, and analyzed flow aggregation. It provides statistical information about the system (popular flows, result distribution, execution time, and so on) and financial information about the return on investment. This API allows you to get the statistic information in order to generate the reports for analyzing information.

Get Statistics

Request: GET /executions/statistics

Description: Returns a flows statistic info (list of FlowStatisticsDataVO): ROI, number of executions, average execution time and result distribution.

Request parameters:

Attribute	Description	Type	Required
top	Get top N results	Integer	No
measurements	Which statistics to display. If nothing is set then the four statistics are displayed. The following options are available: roi, numOfExecutions, avgExecutionTime, resultDistribution.	Predefined Value	No
sortBy	The following options are available: roi,	Predefined	No

	numOfExecutions, avgExecutionTime If nothing is set then: • If the measurements list is empty, then the sort is set to numOfExecutions. • If the Measurements are not empty, then nothing is sorted. If sortBy is set, then it should be contained in measurement (if supplied).	Value	
sortDescending	Default is descending.	Boolean	No
endedBefore	Default is now.	Long	No
endedAfter	Default is one week ago.	Long	No

Example:

GET /executions/statistics?sortBy=avgExecutionTime&endedBefore=1415176455471&top=3

Response status codes:

Code	Meaning	Returned When
200	OK	Operation was successful
400	Bad Request	• Wrong measurements value. • Wrong sortByvalue. It must be included in the measurements, unless it's empty. • Ended after > Ended Before
403	Forbidden	The user does not have dashboard read permission.

Deployment

Deploy Content Packs

Request: PUT /content-packs/{cpFileName}

Description: Deploys a content pack. The file extension should not be provided in the name.

Response status codes:

Code	Meaning	Returned When
201	Created	Deployment succeed
403	Forbidden	The user does not have Manage Content Packs permission.
417	Expectation Failed	Deployment failed

Example:

/content-packs/base-cp

Request path variables:

Attribute	Description	Required
name	The name of the content pack to be deployed.	Yes

Note:

- The body of the request should contain the contents of the content pack file to be deployed (as a raw stream of bytes).

Response entity body:

- on success: Returns a JSON value: true

{

"aggregatedSeverity":"Info",

"contentPackResponses":{

"content pack file name.jar":{

"contentPackUUID":" a4618f99-309d-4537-b67c-e43dd9f73baa",

"contentPackName":"content pack filename.jar",

"message":"content pack file name.jar (author: date:)",

"responses":[

{

"contentPackName":"content pack file name.jar",

"responseCategory":"Success",

"level":"Info",

"message":"Successfully deployed content pack file name.jar"

}

]

}

}

The aggregatedSeverity and level attribute receives one of the following values: Info, Warning, and Error.

The responseCategory attribute receives one of the following values:

- Success: The content pack was deployed successfully.
- ContentPackFile: The content pack file was invalid.
- FlowDependency: Cannot deploy the content pack because of missing flow dependency.
- OperationDependency: Cannot deploy the content pack because of missing operation dependency.
- Overwrite: Cannot deploy the content pack because it can’t overwrite the existed one because of flow/operation dependencies issues.
- ScheduledFlow: A list of scheduled flows that will be affected/deleted if the deployment will be carried out (since the deployment is trying to delete a flow that is scheduled to run).
- Exception: Cannot deploy the content pack because of an unexpected exception.
- on error: Returns a JSON value:

"aggregatedSeverity" : "Error",

"contentPackResponses" :{

"cp.jar": {

"contentPackUUID":"N/A",

"contentPackName":"cp.jar",

"message":"cp.jar (author: , date:)",

"responses":[

{

"contentPackName":"cp.jar",

Develop

"responseCategory":"FlowDependency",

"level":"Error",

"message":"Missing 'operation'/'flow' with UUID d1bbf441-824a-450e-afae-

2ddec0e0f35e, which is required by the 'flow': 'Library/tesdt/flowwww.xml'."

}]

}

}

How can I deploy content packs with progress?

1. Create Deployments – returns deploymentProcessId
2. Upload the Deployment Process File – to the given deploymentProcessId
3. Run a Specific Deployment Process – to the given deploymentProcessId
4. The next step is to request Get the Deployment Process Object. There are two kinds of responses:
    1. "status":"RUNNING" – with data about the progress
    2. "status":"FINISHED" – with data about the result

Create Deployments

Request: POST /deployments

Description: Create a deployment process object and return the id to the client. You can also use this process for deleting content packs.

Response status codes:

Code	Meaning	Returned When
201	Created	Deployment progress object is created.

403	Forbidden	The user does not have Manage Content Packs permission.

JSON response example:

{

"deploymentProcessId":123

}

Upload the Deployment Process File

Request: POST /deployments/{deploymentProcessId}/files

Description: Uploads one or more content pack files, and associates them with an existing deployment process (see Create Deployments). The request should be formatted according to the "multipart/form-data" standard for uploading files to a web server (RFC 2388).

Note: The request body should be the contents of the content pack files to be deployed.

Request path variables:

Attribute	Description
deploymentProcessId	The deployment process ID to associate the files with. The ID is returned when creating a new deployment process.

Example:

POST /deployments/109600004/files

Response status codes:

Code	Meaning	Returned When
200	OK	The files were uploaded and associated with the specified deployment process.
400	Bad Request	Some or all of the uploaded files have failed basic validation.
403	Forbidden	The user does not have Manage Content Packs permission.
404	Not Found	The deployment process object with the specific id does not exist.
409	Conflict	Some or all of the uploaded content packs have already been added to this deployment process.

500	Internal Server Error	The HTTP request was not a valid "multipart/form-data" request (RFC 2388).

JSON response example:

The response body includes details for each of the uploaded files. The signDetails property contains the content pack's digital signature details. The deleteUrl and deleteType properties represent a REST request, which you can send to remove the particular file from this deployment process.

{

"files":

[

{

"name":"cp-signed.jar",

"size":10095,

"fileUploadId":110800009,

"deleteUrl":"/deployments/109600005/files/110800009",

"deleteType":"DELETE",

"signDetails":

{"signStatus":"signed",

"signedBy":"CN=qa-MAINDC-CA, DC=qa, DC=ad, DC=com",

"warnings":[],

"certs":

[

{

"certType":"X.509",

"certDn":"CN=Administrator, CN=Users, DC=qa, DC=ad, DC=com",

"certKeystoreEntryAlias":null,

"validityTimeFrom":1417599979000,

"validityTimeNotAfter":1480758979000,

"validityTimeNotBefore":0,

"supportCodeSign":false

},

{

"certType":"X.509",

"certDn":"CN=qa-MAINDC-CA, DC=qa, DC=ad, DC=com",

"certKeystoreEntryAlias":null,

"validityTimeFrom":1390826569000,

"validityTimeNotAfter":1706359968000,

"validityTimeNotBefore":0,

"supportCodeSign":false

}

],

"trusted":false}

}

]

Run a Specific Deployment Process / Delete Process

Request: PUT /deployments/{deploymentProcessId}

Description: Run the specific deployment process. Make sure that no other process is running.

Request path variables:

Attribute	Description
deploymentProcessId	The requested deployment process id. The id is returned when creating a new deployment process.

Request parameters:

Attribute	Type	Description	Is required?	Default value

force	Boolean	Deploy/delete even when there are warnings (for example, scheduled runs for a flow being deleted).	No	false

Examples:

PUT /deployments/12345?force=true

PUT /deployments/12345

Response status codes:

Code	Meaning	Returned When
204	No Content	The deployment process has started successfully.
403	Forbidden	The user who executed this command does not have permission to manage content packs.
404	Not Found	The requested deployment process is not found.
409	Conflict	Another deployment is in process.
500	Bad Request	The deployment process status is not PENDING or there are no content packs for deployment and no content packs for deleting, or there are both content packs for deployment and content packs for delete. In addition: • When this process is already running or finished. • There are no content pack files for deleting. • Both content pack files for deployment and content files for deleting exist together.

Get the Deployment Process Object

Request: GET /deployments/{deploymentProcessId}

Description: Gets the status and details of a previously-created deployment process. A deployment process is usually used when deploying files with progress, but can also be used for deleting a content pack.

Example:

/deployments/3332190961082830376

Request path variables:

Attribute	Description
deploymentProcessId	The deployment process ID. The ID is returned when creating a new deployment process.

Response status codes:

Code	Meaning	Returned When
200	OK	The specific deployment process was found.
403	Forbidden	The user does not have Read Content Packs or Manage Content Packs permission.
404	Not Found	The deployment process object with the specific id does not exist.

These are the responses:

- status:RUNNING – with data about the progress.
- status:FINISHED – with data about the result.
- status:PENDING – when the deployment was created and files are being uploaded, and before it has started running.

{

"deploymentProcessId": 137100008, "status": "PENDING",

"currentStep": "Downloading files ", "currentStepIndex": 0,

"numOfSteps": 11, "numOfSubSteps": 0, "currentSubStep": 0, "deploymentResultVO": null

}

If your deployment process is still running, the JSON response appears as follows:

{

"deploymentProcessId": 137100005, "status": "RUNNING",

"currentStep": "Deploying artifacts", "currentStepIndex": 12,

"numOfSteps": 16, "numOfSubSteps": 470, "currentSubStep": 409, "deploymentResultVO": null

}

Delete Content Pack from Deployment Process

Request: DELETE /deployments/{deploymentProcessId}/files/{fileId}

Description: Delete a specific content pack from a deployment process.

Response status codes:

Code	Meaning	Returned When
204	No Content	The specific file was deleted from deployment process.
403	Forbidden	The user does not have Manage Content Packs permission.
404	Not Found	The deployment process object with the specific id does not exist.

Deleting Content Packs

To delete a content pack(s), you need to perform the following:

1. Create Deployments, returns the deploymentProcessId

Example: POST /deployments

JSON response example:

```
{
 "deploymentProcessId":12345
}
```

2. Enter the id of the content pack file that you want to delete, to the given deploymentProcessId. See Adding Content Pack Files for Deleting.

Example: POST /deployments/12345/cpsForDelete

3. Run a Specific Deployment Process to the given deploymentProcessId.

Example: PUT /deployments/12345

4. The next step is to request Get the Deployment Process Object.

Adding Content Pack Files for Deleting

Request: POST /deployments/{deploymentProcessId}/cpsForDelete

Description: Add content pack files to the deployment process that you want deleted during the process.

Request path variables:

Attribute	Description
deploymentProcessId	The requested deployment process id. The id is returned when creating a new deployment process.

Example:

POST /deployments/12345/cpsForDelete

Response status codes:

Code	Meaning	Returned When
204	No Content	The requested content packs were added to the deployment process.
403	Forbidden	The user who executed this command does not have permission to manage content packs.
404	Not Found	The requested deployment process is not found.
500	Bad Request	The requested content pack is not deployed or the deployment process status is not PENDING.

Request entity body:

A string array which contains all of the ids of content packs that should be deleted. To acquire the content pack ID see Get Content Packs.

Example:

["39d15573-aad6-44b3-a571-39c98c9bd508","a2c87d2a-2192-4087-8387-763d38246d26"]

Roll Back Last Deployment

Request: DELETE /content-packs/last

Description: Rolls back the last content pack deployment or deletion. Note that only the last action can be rolled back; you cannot roll back twice in a row.

Response status codes:

Code	Meaning	Returned When
204	Successful (no-content)	The last deployed content pack had been removed
400	Bad Request	There was already rollback on the last revision.
403	Forbidden	The user does not have Manage Content Packs permission.
404	Not Found	There is no deployment to rollback.

Content Packs

A content pack is a collection of flows, operations, configuration items (selection lists, system accounts, group aliases, and so on), as well as the binaries required to run actions. A content pack can be created in Studio by an author, or it can be provided by HPE or a third party.

The following table lists the metadata for a content pack:

Attribute	Type	Description	Comments
id	String	The id of the content pack.	
name	String	The name of the content pack.	
version	String	The version of the content pack.	Can be null.
publisher	String	The name of the publisher that created the content pack.	Can be null.
description	String	The description of the content pack.	Can be null.
deploymentDate	Long	Time-stamp in milliseconds when the deployment of the content pack started.	Can be null.

signDetails	SignDetailsVO	The signature details of the content pack	

SignDetailsVO:

Attribute	Type	Description	Comments
signStatus	String	The signature status, can be: signed, notSigned, broken, expired, na	
signedBy	String	The issuer distinguished name of the signature.	
warnings	List<String>	List of signing warnings	Can be empty.
certs	List< CertSignVO>	List of all signature's CAs.	Can be empty.
trusted	boolean	Whether the CA is trusted by OO or not (The CA should be imported into the client.truststore)	

CertSignVO

Attribute	Type	Description	Comments
certType	String	The type of the certificate	
certDn	String	The subject of the certificate	
certKeystoreEntryAlias	String	The CA entry alias	
validityTimeFrom	long	The from date of the certificate validity	milliseconds
validityTimeNotAfter	long	The not after (end) date of the certificate validity	milliseconds
validityTimeNotBefore	long	The not before date of the certificate validity	milliseconds
supportCodeSign	boolean	True if the certificate is designed to be a code signer	

Get Content Packs

Request: GET /content-packs

Description: Retrieves a list of all the deployed content packs and with the related details.

Request parameters:

Attribute	Type	Description	Is required?	Default value

active	Boolean	Allows you to retrieve either a list of all content packs including content (flows, operations and configuration items) and binaries or content packs including just content.	No	False

Example:

GET /content-packs

GET /content-packs?active=false

Response status codes:

Code	Meaning	Returned When
200	OK	The requested content pack was returned.
403	Forbidden	The user who executed this command does not have permission to view content packs.

Response entity body:

An array which contains all of the content packs which are deployed in the central.

Each element in the array represents a content pack. See the content pack meta-data table.

The returned array is ordered by descending deployment times, in other words, the content pack that was deployed last will be first.

Example:

[

{

"name":"cp-with-only-success-flow",

"version":"10.30.2",

"id":"a4deec31-f10e-4249-9a71-7309fe0dbc5a",

"publisher":"OO Tests",

"description":"HPE OO cp-with-only-success-flow Test Content Pack",

"deploymentDate":1418208680626,

"deployedBy":"anonymousUser",

"signDetails":

{

"signStatus":"signed",

"signedBy":"CN=qa-MAINDC-CA, DC=qa, DC=ad, DC=com",

"warnings":[],

"certs":

[

{

"certType":"X.509",

"certDn":"CN=Administrator, CN=Users, DC=qa, DC=ad, DC=com",

"certKeystoreEntryAlias":null,

"validityTimeFrom":1417599979000,

"validityTimeNotAfter":1480758979000,

"validityTimeNotBefore":0,

"supportCodeSign":false

},

{

"certType":"X.509",

"certDn":"CN=qa-MAINDC-CA, DC=qa, DC=ad, DC=com",

"certKeystoreEntryAlias":null,

"validityTimeFrom":1390826569000,

"validityTimeNotAfter":1706359968000,

"validityTimeNotBefore":0,

"supportCodeSign":false

}

],

"trusted":false

}

},

Another one……

]

Get Single Content Pack Details

Request: GET /content-packs/{id}

Description: Retrieves details of the specified deployed content pack.

Request path variables:

Attribute	Description	Type
id	The requested content pack's id. The id may be acquired by retrieving the list of all content packs and extracting the id of element of the required content pack by its name.	int

Example:

GET /content-packs/b137e165-f4f7-4201-b262-2265c8085d27

Response status codes:

Code	Meaning	Returned When
200	OK	The requested content pack was returned.
401	Unauthenticated	The user did not authenticate and anonymous authentication is not allowed.
403	Forbidden	The user who executed this command does not have permission to view content packs.
404	Not Found	The requested content pack was not found.

Response entity body:

A content pack element which represents the details of requested Content Pack. See the content pack meta-data table.

{

"name":"cp-with-only-success-flow",

"version":"10.30.2",

"id":"a4deec31-f10e-4249-9a71-7309fe0dbc5a",

Develop

"publisher":"OO Tests",

"description":"HPE OO cp-with-only-success-flow Test Content Pack",

"deploymentDate":1418208680626,

"deployedBy":"anonymousUser",

"signDetails":

{

"signStatus":"signed",

"signedBy":"CN=qa-MAINDC-CA, DC=qa, DC=ad, DC=com",

"warnings":[],

"certs":

[

{

"certType":"X.509",

"certDn":"CN=Administrator, CN=Users, DC=qa, DC=ad, DC=com",

"certKeystoreEntryAlias":null,

"validityTimeFrom":1417599979000,

"validityTimeNotAfter":1480758979000,

"validityTimeNotBefore":0,

"supportCodeSign":false

},

{

"certType":"X.509",

"certDn":"CN=qa-MAINDC-CA, DC=qa, DC=ad, DC=com",

"certKeystoreEntryAlias":null,

"validityTimeFrom":1390826569000,

"validityTimeNotAfter":1706359968000,

"validityTimeNotBefore":0,

"supportCodeSign":false

}

],

"trusted":false

}

}

Get Content Pack Contents Tree

Request: GET /content-packs/{id}/content-tree

Description: Returns a flat list that of the contents of the content pack in tree-nodes format.

Request path variables:

Attribute	Description
id	The requested content pack's id. The id may be acquired by retrieving the list of all content packs and extracting the id of element of the required content pack by its name.

Example:

GET /content-packs/b137e165-f4f7-4201-b262-2265c8085d27/content-tree

Response status codes:

Code	Meaning	Returned When
200	OK	The requested content pack's contents were returned.
403	Forbidden	The user who executed this command does not have permission to view content packs.
404	Not Found	The requested content pack was not found.

Response entity body:

An array of elements which represents one or two trees that contain all of the entities contained in the content pack.

Each element in the array represents a tree node which is either a folder (which isn't a leaf and has children) or a content element (which is a leaf and doesn't not have children).

The returned array is ordered in the following way: first the tree whose root is the "Library" folder, then the tree whose root is the "Configuration" folder. Each tree is ordered alphabetically by the path attribute.

Attribute	Type	Description	Comments
id	String	The id of node.	For a non-leaf (folder) node, the id is the same as the path. For a leaf node (deployed entity), the id is the entity's id.
name	String	The name of the node.	Folder name or deployed entity name.
parentId	String	The id of the parent node.	Null if this is a root node.
leaf	Boolean	Whether the node is a leaf or not.	
path	String	The path of the node.	
type	Predefined Value	The type of node. Possible values: FOLDER, FLOW, OPERATION, CATEGORY, DOMAIN_ TERM, GROUP_ALIAS, ROLE_ALIAS, SCRIPTLET, SELECTION_LIST, SYSTEM_ACCOUNT, SYSTEM_ EVALUATOR, SYSTEM_FILTER, SYSTEM_PROPERTY	

Get Content Pack Statistics

Request: GET /content-packs/statistics

Description: Retrieves the content pack usage statistics according to the given request parameters.

Notes:

A content pack will be counted if it is actually used by the executed flow (as an operation or subflow). Each content pack will be counted at most once for every execution.

Once an execution ends, it will take 10 to 20 minutes to show up in the statistics.

Request parameters:

Attribute	Type	Description	Required

cpNames	List of Strings	The content pack names to retrieve the statistics for. An empty list means all the content packs. Note that the content pack does not have to be currently deployed.	No
months	List of numbers	The months to retrieve the statistics for. An empty list means all months.	No
years	List of numbers	The years to retrieve the statistics for. An empty list means all years.	No

Example:

GET/content-packs/statistics?cpNames=Base,Solutions&months=1,2,3&years=2015

Response status codes:

Code	Meaning	Returned When
200	OK	The requested content pack statistics was returned.
403	Forbidden	The user who executed this command does not have permission to view or manage content packs.

Response entity body:

A map containing all of the content pack statistics that match the request.

The first level of the map is the statistics years. Each year is mapped to the different months, and each month contains the statistic element. This statistics element contains the total runs for that year and month and a map of content packs with their total runs.

Example:

{
 "2014":{
  "12":{
   "total": 6500,
   "cps":{
    "Base": 6200,
    "Solutions": 1600,
    "Database": 300
   }

Develop


```
  }
 },
 "2015":{
  "1":{
   "total": 7000,
   "cps":{
    "Base": 7000,
    "Solutions": 2000,
    "Database": 1300
   }
  },
  "2":{
   "total": 1000,
   "cps":{
    "Base": 950,
    "Solutions": 400
   }
  }
 }
}
```


Get Content Pack Changes

Request: GET /content-packs/{id}/changes

Description: Retrieves all deployed entities changes from the last deployment of this content pack.

Request path variables:

Attribute	Description

id	The requested content pack's id. The id may be acquired by retrieving the list of all content packs and extracting the id of element of the required content pack by its name.

Example:

GET /content-packs/b32b3a3d-0d7a-4780-85a1-5438987803ef/changes

Response status codes:

Code	Meaning	Returned When
200	OK	The requested content pack's changes were returned.
403	Forbidden	The user who executed this command does not have permission to view content packs.
404	Not Found	The requested content pack was not found.

Response entity body:

An array which contains all of the deployed entities changes from the last deployment of this content pack.

Each element in the array represents a change to a deployed entity. The returned array is not ordered.

Attribute	Type	Description	Comments
id	String	The id of the deployed entity.	
changeType	Predefined Value	The type of change that occurred. Possible values: ADD: The deployed entity was added. DEL: The deployed entity was deleted. MOD: The deployed entity was modified (name changed, moved, xml changed, value changed)	
oldCpName	String	The name of the content pack that the entity belonged to before the last deployment.	Null if the content pack name hasn't changed.
oldPath	String	The path of the entity in the old content pack.	Null if the path has not changed.
currentPath	String	The current path of the entity.	

entityType	Predefined Value	The type of entity that changed. Possible values: FLOW, OPERATION, CATEGORY, DOMAIN_ TERM, GROUP_ALIAS, ROLE_ALIAS, SCRIPTLET, SELECTION_LIST, SYSTEM_ACCOUNT, SYSTEM_ EVALUATOR, SYSTEM_FILTER, SYSTEM_ PROPERTY	
xmlChanged	Boolean	Whether the entity's xml was changed or not.	

Configuration Items

Get a Configuration Item (content)

Request: GET /config-items/{type}/{path}

Description: Returns a Configuration Item, such as a System Account.

Request path variables:

Attribute	Description
type	The type of the Configuration Item to return. Supported types: domain-terms, group-aliases, selection-lists, system-accounts, system-properties
path	The relative path of the Configuration Item. For example, if the item is stored at Configuration/System Accounts/folder/sa.xml, the relative path is folder/sa. Note: Paths are case-sensitive in Central.

Example:

GET /config-items/system-accounts/folder1/folder2/sa1

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Successful
403	Forbidden	The user is not allowed to view Configuration Items
404	Not Found	The Configuration Item does not exist, or the given type is not supported. For System Accounts, this can also mean the user does not have privileges to access this particular item.

Response body:

- on success, returns a JSON object with the following properties:

Attribute	Description	Type
type	The Configuration Item type, in the same format as given in the request URI (see above).	Predefined Value
path	The item's relative path, as given in the request URI (see above)	String
name	The item's name	String
value	The item's actual value, as will be used in flow execution. This will be the same as customValue if set, otherwise it will be the same as defaultValue. • For Domain Terms and Selection Lists, the value is a pipe-delimited collection of values. For example: "First Value|Second Value|Third Value". • For System Accounts, the value is formatted as a JSON object with properties username and password (see example below). Note that the server never sends passwords; they are masked as asterisks.	String
defaultValue	The item's default (deployed) value	String
customValue	The item's custom (override) value, which can be set in Central or through REST	String
sensitive	True when the entire value of the configuration item is hidden	Boolean
fullPath	The item's full path in the Content Pack, including the .xml suffix	String
uuid	The item's UUID – will be null if this item is not deployed (or if it is a duplicate)	String
referencedId	The ID used to reference the configuration item in flows and REST API	String

Example:

{

"type":"system-accounts", "path":"folder1/folder2/sa1", "name":"sa1",

"value":"{\"username\":\"admin\",\"password\":\"**********\"}", "defaultValue":"
{\"username\":\"admin\",\"password\":\"**********\"}", "customValue":null,

"fullPath":"Configuration/System Accounts/sa1.xml", "uuid":"4f32bcb8-969c-470f-9803-f823b72a9436"

}

Get Configuration Items by type (content)

Request: GET /config-items/{type}

Description: Returns all Configuration Items of the specified type.

Request path variables:

Attribute	Description
type	The type of the Configuration Item to return. For details, see Get a Configuration Item (content).

Example:

GET /config-items/system-accounts

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Successful
403	Forbidden	The user is not allowed to view Configuration Items
404	Not Found	The given type is not supported

Response body:

On success, returns a JSON array of Configuration Items. The array will be empty if there are no items of the requested type.

For the format of each item, see Get a Configuration Item (content).

Get All Configuration Items (content)

Request: GET /config-items

Description: Returns all Configuration Items of the supported types (see Get a Configuration Item (content) for a list of these types).

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Successful
403	Forbidden	The user is not allowed to view Configuration Items

Response body:

Returns a JSON array of Configuration Items (will be empty if there are none). For the format of each item, see Get a Configuration Item (content).

Set a Configuration Item's value (content)

Request: PUT /config-items/{type}/{path}

Description: Sets (or clears) the custom value of an existing Configuration Item, such as a System Account. If the item has a default (deployed) value, the custom value will override it.

Request path variables:

Attribute	Description
type	The type of the Configuration Item to return. For details, see Get a Configuration Item (content).
path	The relative path of the Configuration Item. For details, see Get a Configuration Item (content).

Example:

PUT /config-items/system-accounts/folder1/folder2/sa1

Request body:

The body must include the new value as a JSON string. To clear the value, put null in the body.

Examples:

- "some value"
- "some value with \"quotes\" in it"
- null

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Successful
400	Bad Request	The given value is not valid JSON (must be either a string or null), or it is not formatted correctly for this Configuration Item's type
403	Forbidden	The user is not allowed to manage Configuration Items
404	Not Found	The Configuration Item does not exist, or the given type is not supported For System Accounts, this can also mean the user does not have privileges to access this particular item.

Note: Some Configuration Item types expect a particular value format. For details, see the response body section in Get a Configuration Item (content).

Response body:

- on success: returns the updated Configuration Item. See Get a Configuration Item (content) for details on the object format

Delete a Configuration Item (content)

Request: DELETE /config-items/{type}/{path}

Description: Deletes a Configuration Item, such as a System Account. Note that this will fail if the item to delete is currently marked as deployed (in other words, its uuid property is not null).

- To delete a deployed Configuration Item, remove it from the Content Pack and redeploy.
- If you want to clear the override value, see Set a Configuration Item's value (content).

Request path variables:

Attribute	Description
type	The type of the Configuration Item to return.

	For details, see Get a Configuration Item (content).
path	The relative path of the Configuration Item. For details, see Get a Configuration Item (content).

Example:

DELETE /config-items/system-accounts/folder1/folder2/sa1

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Successful
403	Forbidden	The Configuration Item is currently deployed (its uuid property is not null), or the user is not allowed to manage Configuration Items
404	Not Found	The Configuration Item does not exist, or the given type is not supported. For System Accounts, this can also mean the user does not have privileges to access this particular item.

Response body:

On success, returns the deleted Configuration Item. See Get a Configuration Item (content) for details on the object format.

Get Content Configuration tree

Request: GET /config-items/tree

Description: Returns the content configuration tree.

Example:

GET /config-items/tree

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Successful. If there are no results, the result will be empty.

Response entity body:

Develop

[{
"id":"Configuration",
"parentId":null,
"leaf":false,
"path":"Configuration",
"name":"Configuration",
"type":"FOLDER",
"icon":null

},

{
"id":"Configuration/Group Aliases",
"parentId":"Configuration",
"leaf":false,
"path":"Configuration/Group Aliases",
"name":"Group Aliases",
"type":"FOLDER",
"icon":null

},

{
"id":"Configuration/Group Aliases/RAS_Operator_Path.xml",
"parentId":"Configuration/Group Aliases",
"leaf":true,
"path":"Configuration/Group Aliases/RAS_Operator_Path.xml",

"name":"RAS_Operator_Path",

"type":"GROUP_ALIAS",

"icon":null,

"workerGroup":"RAS_Operator_Path",

"deployed":true

},

{

"id": "Configuration/System Accounts/central_user.xml",

"parentId": "Configuration/System Accounts",

"leaf": true,

"path": "Configuration/System Accounts/central_user.xml",

"name": "central_user",

"type": "SYSTEM_ACCOUNT",

"icon": null,

"deployedValue": "sa-change-it-case",

"overrideValue": null,

"sensitive": false,

"deployed": true,

"referencedId": "central_user"

},

{

"id": "Configuration/System Properties/rerunSystemProperty.xml",

"parentId": "Configuration/System Properties",

"leaf": true,

"path": "Configuration/System Properties/rerunSystemProperty.xml",

"name": "rerunSystemProperty",

Develop


```
  "type": "SYSTEM_PROPERTY",
  "icon": null,
  "deployedValue": "************",
  "overrideValue": null,
  "sensitive": true,
  "deployed": true,
  "referencedId": "rerunSystemProperty"
 }
]
```


Get Configuration Item Details

Request: GET /config-items/{type}/{path}?details=true

Description: Returns the deployed entity data of a configuration item

Example:

GET /config-items/system-accounts/folder1/folder2/sa1?details=true

Request path variables:

Attribute	Description
type	The type of the Configuration Item to return. For details, see Get a Configuration Item (content).
path	The relative path of the Configuration Item. For details, see Get a Configuration Item (content).

Request parameters:

This API requires the parameter "details=true", otherwise it will be treated as Get a Configuration Item (content). There are no other parameters.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested path was found.
403	Forbidden	The user is not allowed to view Configuration Items
404	Not Found	The Configuration Item does not exist, or the given type is not supported. For System Accounts, this can also mean the user does not have privileges to access this particular item

Response entity body:

- on success: Returns a JSON object with the following format:

{

"id": "2283acc2-95f0-4cc4-8f45-e5053ace7aaf",

"name": "some-prop",

"path": "Configuration/System Properties/some-prop.xml",
"description": "This is just some property",

"cpName": "cp-props",

"version": "10.20.01 "

}

Audit

HPE OO now gives you the option to audit events, so that you can track security breaches. Auditing lets you track actions that took place on Central, such as logins, triggering flows, creating schedules, editing configurations, and so on.

Get Audit Configuration

Request: GET /audit/config

Description: Gets the audit configuration.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested contents were returned.
403	Forbidden	The user attempting to execute this command does not have the View Security Configuration or the Manage Security Configuration permissions.

Response entity body:

Attribute	Type	Description	Comment
enabled	Boolean	True when auditing is enabled. False when auditing is disabled.	

Example:

{"enabled":true}

Update Audit Configuration

Request: PUT /audit/config

Description: Updates the audit configuration.

Request entity body:

Attribute	Type	Description	Required	Default
enabled	Boolean	True to enable auditing. False to disable auditing.	Yes	

Example:

{"enabled":true}

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The audit configuration was updated successfully.
400	Bad Request	The data in the request body is invalid.
403	Forbidden	The user attempting to execute this command does not have the Manage Security Configuration permission.

Response entity body:

Returns the updated audit configuration object.

Example:

{"enabled":true}

Get Audit Records

Request: GET /audit/records

Description: Returns a page of the audit records that were recorded in reverse chronological order.

Request parameters:

Attribute	Type	Description	Required	Default
sortDescending	Boolean	Whether to sort by descending order. If false – sort by ascending order.	No	true
pageNum	Integer	The number of the returned page. This must be a positive (>0) number.	No	1
pageSize	Integer	The number of records in the returned page. This must be a positive (>0) number.	No	50

Example:

GET /audit/records?sortDescending=false&pageNum=2&pageSize=10

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested contents were returned.
400	Bad Request	If any of the arguments are invalid.
403	Forbidden	The user attempting to execute this command does not have the View Audit permission.

Response entity body:

An array which contains all of the events which were audited and meet the filtering criteria. Each element in the array represents an audited event.

Attribute	Type	Description	Comment
time	Long	The audit time	
type	Predefined Type	The type of operation being audited. Possible values: See the Audit Types/Groups table.	
group	Predefined Type	The group to which the operation being audited belongs. Possible values: See the Audit Types/Groups table.	
subject	String	The user that performed the operation being audited. If the operation is a general system event, the subject will be the default user name (anonymousUser).	
outcome	Predefined Type	The outcome of the operation being audited. Possible values: Success, Failure and System Error	
data	String	The extra audit data specific to the type of operation being audited.	In JSON key/value format

Delete Audit Records

Request: DELETE /audit/records

Description: Delete the audit records according to the request parameters. Returns the number of records that were actually deleted.

Request parameters:

Attribute	Type	Description	Required	Default
timeBefore	Long	Records of operations that were audited before this time will be purged. In UTC time format.	Yes	
maxAmount	Integer	The maximum amount of records to delete. Note: In some cases the number of records deleted might be slightly larger than the maxAmount that was passed.	No	1000

Example:

DELETE /audit/records?timeBefore=1387218013000&maxAmount=3000

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested contents were deleted.
400	Bad Request	If any of the arguments are invalid.
403	Forbidden	The user attempting to execute this command does not have the Manage Data Cleanup permission

Response entity body:

- on success: Returns the number of audit records that have been deleted.

Note: This number might be higher than the maxAmount parameter that was passed.

Audit Types/Groups

Type	Group
CentralStartup	CentralLifecycle
CentralShutdown	CentralLifecycle
AuditConfigurationChange	AuditManagement
AuthenticationFailure	Authentication-Authorization
AuthorizationFailure	Authentication-Authorization
LoginAttempt	Authentication-Authorization
Logout	Authentication-Authorization
AuthenticationConfigurationUpdate	Authentication-Authorization
CaptureCredentialsConfigurationUpdate	Authentication-Authorization
RoleCreate	Authentication-Authorization
RoleUpdate	Authentication-Authorization
RolesDelete	Authentication-Authorization
RoleSetDefault	Authentication-Authorization
LDAPConfigurationCreate	Authentication-Authorization
LDAPConfigurationUpdate	Authentication-Authorization
LDAPConfigurationsDelete	Authentication-Authorization

Type	Group
InternalUserCreate	Authentication-Authorization
InternalUserUpdate	Authentication-Authorization
InternalUsersDelete	Authentication-Authorization
SAMLConfigurationCreate	Authentication-Authorization
SAMLConfigurationUpdate	Authentication-Authorization
SAMLConfigurationDelete	Authentication-Authorization
SSOConfigurationUpdate	Authentication-Authorization
PathEntitlementUpdate	Authentication-Authorization
RunTriggered	Runs
RunStatusChange	Runs
ScheduleCreate	Runs
ScheduleEdit	Runs
SchedulesEnable	Runs
SchedulesDisable	Runs
SchedulesDelete	Runs
DeploymentProcessCreate	ContentDeployment
ContentUploadToDeploymentProcess	ContentDeployment
ContentRemoveFromDeploymentProcess	ContentDeployment
ContentForDeleteAddToDeploymentProcess	ContentDeployment
DeploymentProcessStart	ContentDeployment
ContentDeployment	ContentDeployment
ContentRollback	ContentDeployment
ContentDelete	ContentDeployment
GroupAliasCreate	ContentConfiguration
GroupAliasUpdate	ContentConfiguration
GroupAliasesDelete	ContentConfiguration
SystemAccountCreate	ContentConfiguration

Type	Group
SystemAccountUpdate	ContentConfiguration
SystemAccountsDelete	ContentConfiguration
ContentConfigurationItemCreate	ContentConfiguration
ContentConfigurationItemUpdate	ContentConfiguration
ContentConfigurationItemDelete	ContentConfiguration
WorkerRegister	TopologyManagement
WorkersDelete	TopologyManagement
WorkerUpdate	TopologyManagement
WorkersUpdate	TopologyManagement
WorkersAssignToGroup	TopologyManagement
WorkersRemoveFromGroup	TopologyManagement
ExternalUrlCreateOrUpdate	TopologyManagement
ExternalUrlDelete	TopologyManagement
SystemConfigurationCreateOrUpdate	SystemConfiguration
SystemConfigurationUpdate	SystemConfiguration
SystemConfigurationDelete	SystemConfiguration
FlowRerun	Runs
RerunDataPurging	Runs
RunDataPurging	Runs
AddReverseRAS	TopologyManagement
UpdateReverseRAS	TopologyManagement
ForwardProxyCreateOrUpdate	TopologyManagement
ForwardProxyDelete	TopologyManagement
TestConnectionToReverseRAS	TopologyManagement
RasUpgradeUploadFile	TopologyManagement
RasUpgradeDeleteFile	TopologyManagement
RasUpgradePrepare	TopologyManagement
RasUpgradeStart	TopologyManagement

LDAP Configuration

The LDAP API allows you to configure you organization's LDAP.

This enables users to log in with their organizational credentials and for the administrator to map LDAP groups to OO Roles.

The LDAP API includes a test API to verify configurations are going to be set correctly before saving them.

Note: It is recommended to set LDAP configurations when you want to authenticate users and not rely on the internal users feature, which are less secure.

Although with the LDAP API the configurations are set, you should enable the system authentication for them to take place.

In the case both the LDAP configurations and internal users were set, the LDAP settings override the internal user settings, if there is a collision between user IDs.

API supported with multiple LDAPs

If your organization works with multiple LDAP servers, it is now possible to configure Central to work with all of them. This includes LDAPs with different schemes and from different vendors. For example, you might have an Active Directory (Microsoft LDAP) implementation for one part of the organization and a Sun One (Oracle LDAP) implementation for another part.

Get LDAP Configuration by ID

Request: GET /authns/ldap-config/{id}

Description: Retrieves an LDAP configuration according to the specified ID.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	LDAP was retrieved successful.
403	Forbidden	User does not have view/manage security configurations permissions.
404	Not found	The requested LDAP does not exist.

Response entity body:

Attribute	Type	Description
id	String	The ID for the returned LDAP.
type	Predefined Value	The type of the LDAP. See the LDAP Appendix for more information.
domain	String	The domain of the LDAP.
addresses	An array of Address	The address of the LDAP and other addresses for the case of failover. See the LDAP Appendix for more information.
securedChannel	boolean	True if a secured channel is used (SSL).
enabled	boolean	Whether the returned LDAP is operational.
privilegedUser	String	The DN of a user with search capabilities on the provided User & Group DNs. In case the LDAP type is ACTIVE_DIRECTORY the exact user name will be returned (and not a DN).
privilegedUserPassword	String	Indicates whether a password for the privileged user was provided. "********" appears when a password was set.
groupsDns	String[]	The DNs on which to apply the groups filter for search.
groupsSearchRecursive	boolean	Whether groups search should be recursive.
groupsFilter	String	A search filter to apply on the groups DNs.
usersDns	String[]	The DNs on which to apply the users filter for search.
usersSearchRecursive	boolean	Whether users search should be recursive.
userCommonNameAttribute	String	The attribute of the user which should be used for display purposes.
userEmailAttribute	String	The attribute of the user which contains the mail address.
groupMembershipAttribute	String	In ACTIVE_DIRECTORY type, this represents the attribute of the user which contains the groups. In any other type, it is null.
groupNameAttribute	String	The attribute of the group which contains the group's name used for mapping Roles to Groups.
usersFilter	String	A search filter to apply on the users DNs.

Attribute	Type	Description
customGroupsAttributesNames	String	Attribute names that will be used as groups. Separated by semicolon.

Example

{

"id":"72cdc1a7-1005-4800-a412-5e4a9b8f6bec", "type":"OPEN_DJ",

"domain":"Indigo", "addresses":[

{"port":389,"host":"mysite.com"},

{"port":389,"host":"mysite.com"},

{"port":3021,"host":"mysite.com"}

], "securedChannel":false, "enabled":true,

"privilegedUser":"uid=user,ou=people,dc=HPE,dc=com", "privilegedUserPassword":"********",

"groupsDns":[

"ou=products,dc=HPE,dc=com"

], "groupsSearchRecursive":true, "groupsFilter":"(uniqueMember={0})", "groupNameAttribute":"cn", "usersFilter":"(&(objectclass=person)(uid={0}))", "usersDns":[

"ou=people,dc=HPE,dc=com"

], "usersSearchRecursive":true, "userCommonNameAttribute":"cn", "userEmailAttribute":"mail", "groupMembershipAttribute":null "customGroupsAttributesNames":null

}

Create a New LDAP Configuration

Request: POST /authns/ldap-config

Description: Creates an LDAP configuration.

Response status codes:

Code	Meaning	Returned When
201	Successful (Created)	LDAP was successfully created

400	Bad Request	Client didn't provide the mandatory field or provided some fields with wrong format.
403	Forbidden	User does not have manage security configuration permissions.
409	Conflict	The provided domain already exists.

Request body:

Attribute	Type	Description	Required?	Default value
type	Predefined Value	The type of LDAP. See the LDAP Appendix for more information.	No	OTHER
domain	String	The domain of the LDAP. For Active Directory this must be a real domain that can be bound on. In all other cases any name is suitable.	Yes	
addresses	An array of Address	The address of the LDAP and other addresses for the case of fail-over. See the LDAP Appendix for more information.	Yes	
securedChannel	boolean	True if a secured channel should be used (SSL).	No	False
enabled	boolean	Whether the provided LDAP should be operational.	No	False
privilegedUser	String	The DN of a user with search capabilities on the provided User and Group DNs. In case the	No	None

Attribute	Type	Description	Required?	Default value
		LDAP type is ACTIVE_ DIRECTORY the exact user name should be provided (and not a DN). If the LDAP allows anonymous search, this field is not required.		
privilegedUserPassword	String	The password of the privileged user.	No	None
groupsDns	String[]	The DNs on which to apply the groups filter for search.	Required for types other than ACTIVE_ DIRECTORY and ALTERNATE_ GROUP	
groupsSearchRecursive	boolean	Whether groups search should be recursive	No	False
groupsFilter	String	A search filter to apply on the groups DNs.	Required for types other than ACTIVE_ DIRECTORY and ALTERNATE_ GROUP	
usersDns	String[]	The DNs on which to apply the users filter for search.	Yes	
usersSearchRecursive	boolean	Whether users search should be recursive	No	False
userCommonNameAttribute	String	The attribute of the user which should be used for display purposes.	Yes	
userEmailAttribute	String	The attribute of the	No	None

Attribute	Type	Description	Required?	Default value
		user which contains the mail address		
groupMembershipAttribute	String	In ACTIVE_ DIRECTORY type, this represents the attribute of the user which contains the groups. In any other type, it is null.	Required only for ACTIVE_ DIRECTORY type.	Mandatory for ACTIVE_ DIRETORY
groupNameAttribute	String	The attribute of the group which contains the group's name used for mapping Roles to Groups.	Required for all types other than ALTERNATE_ GROUP.	
usersFilter	String	A search filter to apply on the users DNs.	Yes	
customGroupsAttributesNames	String	Attribute names that will be used as groups. Separated by semicolon.	Required only for ALTERNATE_ GROUP type	

Get All LDAP Configurations

Request: GET /authns/ldap-config

Description: Retrieves all the configured LDAPs.

Response status codes:

Code	Meaning	Returned When
200	OK	LDAPs were retrieved successfully.
403	Forbidden	User does not have view/manage security configurations permissions.

Response entity body:

An array of LDAP configurations. An empty array if none exists.

Attribute	Type	Description
id	String	The ID for the returned LDAP.
type	Predefined Value	The type of the LDAP. See the LDAP Appendix for more information.
domain	String	The domain of the LDAP.
addresses	An array of Address	The address of the LDAP and other addresses for the case of failover. See the LDAP Appendix for more information.
securedChannel	boolean	True if a secured channel is used (SSL).
enabled	boolean	Whether the returned LDAP is operational.
privilegedUser	String	The DN of a user with search capabilities on the provided User & Group DNs. In case the LDAP type is ACTIVE_DIRECTORY the exact user name will be returned (and not a DN).
privilegedUserPassword	String	Indicates whether a password for the privileged user was provided. "*********" is displayed when a password is set.
groupsDns	String[]	The DNs on which to apply the groups filter for search.
groupsSearchRecursive	boolean	Whether groups search should be recursive.
groupsFilter	String	A search filter to apply on the groups DNs.
usersDns	String[]	The DNs on which to apply the users filter for search.
usersSearchRecursive	boolean	Whether users search should be recursive.
userCommonNameAttribute	String	The attribute of the user which should be used for display purposes.
userEmailAttribute	String	The attribute of the user which contains the mail address.
groupMembershipAttribute	String	In ACTIVE_DIRECTORY type, this represents the attribute of the user which contains the groups. In any other type, it is null.
usersFilter	String	A search filter to apply on the users DNs.
groupNameAttribute	String	The attribute of the group which contains the group's name used for mapping Roles to Groups.

customGroupsAttributesNames	String	Attribute names that will be used as groups. Separated by semicolon.

Get Default LDAP

Request: GET /authns/ldap-config/default

Description: Retrieve the default LDAP.

Response status codes:

Code	Meaning	Returned When
200	OK	LDAPs were retrieved successfully.
403	Forbidden	User does not have read/manage security configuration permissions.
404	Not Found	A default LDAP does not exist.

Response entity body:

Attribute	Type	Description
id	String	The ID for the returned LDAP.
type	Predefined Value	The type of the LDAP. See the LDAP Appendix for more information.
domain	String	The domain of the LDAP.
addresses	An array of Address	The address of the LDAP and other addresses for the case of failover. See the LDAP Appendix for more information.
securedChannel	boolean	True if a secured channel is used (SSL).
enabled	boolean	Whether the returned LDAP is operational.
privilegedUser	String	The DN of a user with search capabilities on the provided User & Group DNs. In case the LDAP type is ACTIVE_DIRECTORY the exact user name will be returned (and not a DN).
privilegedUserPassword	String	Indicates whether a password for the privileged user was provided. "********" is displayed when a password is set.
groupsDns	String[]	The DNs on which to apply the groups filter for search.

Attribute	Type	Description
groupsSearchRecursive	boolean	Whether groups search should be recursive.
groupsFilter	String	A search filter to apply on the groups DNs.
Attribute	Type	Description
usersDns	String[]	The DNs on which to apply the users filter for search.
usersSearchRecursive	boolean	Whether users search should be recursive.
userCommonNameAttribute	String	The attribute of the user which should be used for display purposes.
userEmailAttribute	String	The attribute of the user which contains the mail address.
groupMembershipAttribute	String	In ACTIVE_DIRECTORY type, this represents the attribute of the user which contains the groups. In any other type, it is null.
customGroupsAttributesNames	String	Attribute names that will be used as groups. Separated by semicolon.
groupNameAttribute	String	The attribute of the group which contains the group's name used for mapping Roles to Groups.
usersFilter	String	A search filter to apply on the users DNs.

Get All LDAP Configurations

Request: GET /authns/ldap-config

Description: Retrieves all the configured LDAPs.

Response status codes:

Code	Meaning	Returned When
200	OK	LDAPs were retrieved successfully.
403	Forbidden	User does not have view/manage security configurations permissions.

Response entity body:

An array of LDAP configurations. An empty array if none exists.

Attribute	Type	Description
id	String	The ID for the returned LDAP.
type	Predefined Value	The type of the LDAP. See the LDAP Appendix for more information.
domain	String	The domain of the LDAP.
addresses	An array of Address	The address of the LDAP and other addresses for the case of failover. See the LDAP Appendix for more information.
securedChannel	boolean	True if a secured channel is used (SSL).
enabled	boolean	Whether the returned LDAP is operational.
privilegedUser	String	The DN of a user with search capabilities on the provided User & Group DNs. In case the LDAP type is ACTIVE_DIRECTORY the exact user name will be returned (and not a DN).
privilegedUserPassword	String	Indicates whether a password for the privileged user was provided. "*********" is displayed when a password is set.
groupsDns	String[]	The DNs on which to apply the groups filter for search.
groupsSearchRecursive	boolean	Whether groups search should be recursive.
groupsFilter	String	A search filter to apply on the groups DNs.
usersDns	String[]	The DNs on which to apply the users filter for search.
usersSearchRecursive	boolean	Whether users search should be recursive.
userCommonNameAttribute	String	The attribute of the user which should be used for display purposes.
userEmailAttribute	String	The attribute of the user which contains the mail address.
groupMembershipAttribute	String	In ACTIVE_DIRECTORY type, this represents the attribute of the user which contains the groups. In any other type, it is null.
usersFilter	String	A search filter to apply on the users DNs.
groupNameAttribute	String	The attribute of the group which contains the group's name used for mapping Roles to Groups.

customGroupsAttributesNames	String	Attribute names that will be used as groups. Separated by semicolon.

Get Default LDAP

Request: GET /authns/ldap-config/default

Description: Retrieve the default LDAP.

Response status codes:

Code	Meaning	Returned When
200	OK	LDAPs were retrieved successfully.
403	Forbidden	User does not have read/manage security configuration permissions.
404	Not Found	A default LDAP does not exist.

Response entity body:

Attribute	Type	Description
id	String	The ID for the returned LDAP.
type	Predefined Value	The type of the LDAP. See the LDAP Appendix for more information.
domain	String	The domain of the LDAP.
addresses	An array of Address	The address of the LDAP and other addresses for the case of failover. See the LDAP Appendix for more information.
securedChannel	boolean	True if a secured channel is used (SSL).
enabled	boolean	Whether the returned LDAP is operational.
privilegedUser	String	The DN of a user with search capabilities on the provided User & Group DNs. In case the LDAP type is ACTIVE_DIRECTORY the exact user name will be returned (and not a DN).
privilegedUserPassword	String	Indicates whether a password for the privileged user was provided. "********" is displayed when a password is set.
groupsDns	String[]	The DNs on which to apply the groups filter for search.

Attribute	Type	Description
groupsSearchRecursive	boolean	Whether groups search should be recursive.
groupsFilter	String	A search filter to apply on the groups DNs.
Attribute	Type	Description
usersDns	String[]	The DNs on which to apply the users filter for search.
usersSearchRecursive	boolean	Whether users search should be recursive.
userCommonNameAttribute	String	The attribute of the user which should be used for display purposes.
userEmailAttribute	String	The attribute of the user which contains the mail address.
groupMembershipAttribute	String	In ACTIVE_DIRECTORY type, this represents the attribute of the user which contains the groups. In any other type, it is null.
customGroupsAttributesNames	String	Attribute names that will be used as groups. Separated by semicolon.
groupNameAttribute	String	The attribute of the group which contains the group's name used for mapping Roles to Groups.
usersFilter	String	A search filter to apply on the users DNs.

Delete Default LDAP Marking

Request: DELETE /authns/ldap-config/default

Description: Deletes the default LDAP marking.

Note: The LDAP is not deleted.

Response status codes:

Code	Meaning	Returned When
204	No Content	No default LDAP marking exists anymore
403	Forbidden	User does not have manage security configuration permissions.

Create or Update a Default LDAP Marking

Request: PUT/authns/ldap-config/default or POST/authns/ldap-config/default

Description: Creates or updates default LDAP marking.

Request body:

Attribute	Type	Description	Required?	Default value
id	String	The ID that represents the LDAP to be marked as default	Yes	N/A

Example:

{

"id":"72cdc1a7-1005-4800-a412-5e4a9b8f6bec"

}

Response status codes:

Code	Meaning	Returned When
200	OK	The default LDAP was set accordingly. There was a default LDAP before.
201	Created	The default LDAP was set accordingly. There was no default LDAP before.
400	Bad Request	The user did not provide a mandatory attribute, or provided an invalid value, for example, the given ID does not correspond with an existing LDAP configuration.
403	Forbidden	User does not have manage security configuration permissions.
404	Not Found	The provided ID does not relate to some existing LDAP

Response entity body:

Attribute	**Type**	**Description**
id	String	The ID for the returned LDAP.
type	Predefined Value	The type of the LDAP. See the LDAP Appendix for more information.

Attribute	Type	Description
domain	String	The domain of the LDAP.
addresses	An array of Address	The address of the LDAP and other addresses for the case of failover. See the LDAP Appendix for more information.
securedChannel	boolean	True if a secured channel is used (SSL).
enabled	boolean	Whether the returned LDAP is operational.
privilegedUser	String	The DN of a user with search capabilities on the provided User & Group DNs. In case the LDAP type is ACTIVE_DIRECTORY the exact user name will be returned (and not a DN).
privilegedUserPassword	String	Indicates whether a password for the privileged user was provided. Is displayed "********"when a password is set.
groupsDns	String[]	The DNs on which to apply the groups filter for search.
groupsSearchRecursive	boolean	Whether groups search should be recursive.
groupsFilter	String	A search filter to apply on the groups DNs.
usersDns	String[]	The DNs on which to apply the users filter for search.
usersSearchRecursive	boolean	Whether users search should be recursive.
userCommonNameAttribute	String	The attribute of the user which should be used for display purposes.
userEmailAttribute	String	The attribute of the user which contains the mail address.
groupMembershipAttribute	String	In ACTIVE_DIRECTORY type, this represents the attribute of the user which contains the groups. In any other type, it is null.
groupNameAttribute	String	The attribute of the group which contains the group's name used for mapping Roles to Groups.
customGroupsAttributesNames	String	Attribute names that will be used as groups. Separated by semicolon.
usersFilter	String	A search filter to apply on the users DNs.

Testing LDAP Configurations

There are two APIs for testing LDAP configurations:

- Request: POST /authns/ldap-config/{id}/test

Description: Test an existing LDAP configuration.

- Request: POST /authns/ldap-config/test

Description: Test an ad hoc LDAP configuration.

Request body (only for ad hoc API):

Attribute	Type	Description	Required?	Default value
userName	String	The user to test the provided configurations with.	Yes	
userPassword	String	The password of the user. In case the password is not provided, there will be no authentication attempt.	No	None
type	Predefined Value	The type of the LDAP. See the LDAP Appendix for more information.	No	OTHER
domain	String	The domain of the LDAP. In case of Active Directory this must be a real domain that can be bind on, in all other cases any name would fit.	Yes	
addresses	An array of Address	The address of the LDAP and other addresses for the case of fail-over.	Yes	See the LDAP Appendix

Attribute	Type	Description	Required?	Default value
				for more information
securedChannel	boolean	True if a secured channel should be used (SSL)	No	False
privilegedUser	String	The DN of a user with search capabilities on the provided User & Group DNs. In case the LDAP type is ACTIVE_ DIRECTORY the exact user name should be provided (and not a DN). If the LDAP allows anonymous search, this field is not required.	No	None
privilegedUserPassword	String	The password of the privileged user. If the password of the existing LDAP should be used, displayed with asterisk ("********"). Note: If you decided to provide the password itself, the /authns/ldap-config/test is more suitable. See Ad-hoc Flow Execution.	No	None
groupsDns	String[]	The DNs on which to apply the groups filter for search.	Yes	
groupsSearchRecursive	boolean	Whether groups search should be recursive	No	False

Attribute	Type	Description	Required?	Default value
groupsFilter	String	A search filter to apply on the groups DNs	Yes	
usersDns	String[]	The DNs on which to apply the users filter for search.	Yes	
usersSearchRecursive	boolean	Whether users search should be recursive	No	False
userCommonNameAttribute	String	The attribute of the user which should be used for display purposes.	Yes	
userEmailAttribute	String	The attribute of the user which contains the mail address	No	None
groupMembershipAttribute	String	In ACTIVE_ DIRECTORY type, this represents the attribute of the user which contains the groups. In any other type, it is null.	No	Mandatory for ACTIVE_ DIRETORY
groupNameAttribute	String	The attribute of the group which contains the group's name used for mapping Roles to Groups.	Required for all types other than ALTERNATE_ GROUP	
usersFilter	String	A search filter to apply on the users DNs.	Yes	
enabled	boolean	Whether the returned LDAP is operational	No	False
customGroupsAttributesNames	String	Attribute names that will be used as groups. Separated by semicolon.	Required only for ALTERNATE_ GROUP type	

Response status codes:

Code	Meaning	Returned When
200	OK	The provided LDAP configurations were legal and a test was performed.
400	Bad Request	The provided configurations were bad or server encountered some error while performing the test.
403	Forbidden	User does not have read/manage security configuration permissions.
404	Not found	The provided LDAP id does not exist.

Response body:

Attribute	Type	Description	Comments
authenticated	boolean	Whether or not the user was authenticated.	
groupsNames	String[]	An array with the names of the groups the user belongs to.	
userAttributes	Attribute []	An array with the attributes of the user.	See the LDAP Appendix for more information.
multipleResults	boolean	Whether or not multiple users with the given name were found.	

- RestError for Bad Request (400 code). See the RestError table for more information.

Get LDAP root Details

Request: POST /authns/ldap-root or PUT /authns/ldap-root

Description: Retrieves LDAP root details.

Request body:

Attribute	Type	Description	Required?	Default value
addresses	Address []	An array of addresses. See the LDAP Appendix for more information.	Yes	
securedChannel	boolean	Whether to establish a secured connection (SSL)	No	False

Example:

{"addresses":[

{"host":"mydwbld0006g.isr.HPE.com","port":389},

{"host":"mydwbld0126g.isr.HPE.com","port":3089}

], "securedChannel":false

}

Response status codes:

Code	Meaning	Returned When
200	OK	LDAPs were retrieved successfully
400	Bad Request	Client provided illegal/incorrect parameters.

Successful response body:

Attribute	Type	Description	Comments
rootDns	String[]	An array of DNs which represents the LDAProots.	
serverAddress	Address	The address to which OO successfully connected. See the LDAP Appendix for more information.	
vendorName	String	The name of the vendor of this LDAP	Not all LDAPs support this functionality
vendorVersion	String	The version of this LDAP	Not all LDAPs support this functionality

Example:

{ "rootDns":["dc=mercury,dc=com", "dc=indigo,dc=com"

], "serverAddress":

{"port":389,"host":"mysite.com"}, "vendorName":"ForgeRock AS.", "vendorVersion":"OpenDJ 2.6.0"

}

- Bad Request (400 Code) response. See the RestError table for more information.

Update an Existing LDAP Configuration

Request: PUT /authns/ldap-config/{ldapId}

Description: Update an existing LDAP configuration

Request body:

Attribute	Type	Description	Required?	Default value
type	Predefined Value	The type of LDAP. See the LDAP Appendix for more information.	Yes	
domain	String	The domain of the LDAP. For Active Directory this must be a real domain that can be bound on. In all other cases any name is suitable.	Yes	
addresses	An array of Address	The address of the LDAP and other addresses for the case of fail-over. See the LDAP Appendix for more information.	Yes	
securedChannel	boolean	True if a secured channel should be used (SSL).	No	False
enabled	boolean	Whether the provided LDAP should be operational.	No	False
privilegedUser	String	The DN of a user with search capabilities on the provided User and Group DNs. In case the LDAP type is ACTIVE_ DIRECTORY the exact user name should be provided (and not a DN). If the LDAP allows anonymous search, this field is not required.	No	None
privilegedUserPassword	String	The password of the privileged user.	No	None
groupsDns	String[]	The DNs on which to apply the groups filter for search.	Yes	

groupsSearchRecursive	boolean	Whether groups search should be recursive	No	False
groupsFilter	String	A search filter to apply on the groups DNs	Yes	
usersDns	String[]	The DNs on which to apply the users filter for search.	Yes	
usersSearchRecursive	boolean	Whether users search should be recursive	No	False
userCommonNameAttribute	String	The attribute of the user which should be used for display purposes.	Yes	
userEmailAttribute	String	The attribute of the user which contains the mail address	No	None
groupMembershipAttribute	String	In ACTIVE_ DIRECTORY type, this represents the attribute of the user which contains the groups. In any other type, it is null.	No	Mandatory for ACTIVE_ DIRETORY

Response status codes:

Code	Meaning	Returned When
200	Successful	The LDAP configuration was updated successfully.
400	Bad Request	Client didn't provide the mandatory field or provided some fields with wrong format.
403	Forbidden	User does not have manage security configuration permissions.
409	Conflict	The provided domain already exists.

Example:

{

"type":"SUN_ONE", "domain":"MyDomain",

"addresses":[{"host":"mysite.com","port":389}], "securedChannel":false,

"enabled":true, "privilegedUser":"uid=user,ou=people,dc=HPE,dc=com",
"privilegedUserPassword":"1234", "groupsDns":["ou=products,dc=HPE,dc=com"],
"groupsSearchRecursive":true, "groupsFilter":"(uniqueMember={0})", "groupNameAttribute":"cn",

"usersFilter":"(&(objectclass=person)(uid={0}))", "usersDns":["ou=people,dc=HPE,dc=com"], "usersSearchRecursive":true, "userCommonNameAttribute":"cn"

}

Workers

Many deployments can benefit from having more than a single Worker in a specific environment. For example, this could be helpful if you are managing a remote data center in which you need Workers to be able to withstand the action execution load, or simply for high availability of the Workers in that data center. In previous versions, a load balancer would have been required to balance the load between two Workers, which Central would know as a single logical Worker. See the *Concepts Guide* Key Concepts for more information.

Get All RASes

Request: GET /rases

Description: Retrieves the information of all the RASes in the system.

Response status codes:

Code	Meaning	Returned When
200	OK	Successful
403	Forbidden	The user does not have the View Topology or Manage Topology permission.

Response entity body:

- on success: Returns a JSON object with the following format:

[

{

"upgradeStatus": "READY_FOR_UPGRADE",

"upgradeDetails": null,

"upgradeVersion":null,

"upgradeDetailsForLoaclization":null,

"worker":{

"uuid": "1fc88b1f-33f3-4178-a3ee-996f90f36feb",

"installPath": "C:\\PROJS\\INSTALLATIONS\\10_60\\ras",

"os": "Windows 8.1",

"jvm": "1.8.0_45",

"description": "1fc88b1f-33f3-4178-a3ee-996f90f36feb",

"dotNetVersion": "4.x",

"hostName": "somehost",

"groups":["RAS_Operator_Path"],

"status": "RECOVERED",

"executionStatus": "Idle",

"monitorInfo": null,

"ipAddress": null,

"listenPort": 0,

"protocolType": null,

"reversedWorker": false,

“connectedCentralUuid": null,

"connectedCentralHostname":null,

"sharedSecret": null,

"version": "10.70",

"versionId": "20160300",

"active": false

}

}

]

Prepare RAS Upgrade and Start RAS Upgrade

Request: POST /rases/{uuids}/upgrade

Description: Initiates an upgrade action (either prepare upgrade or start upgrade) for the specified RASes.

Response status codes:

Code	Meaning	Returned When
200	OK	The action was initiated for all specified RASes.
403	Forbidden	The user does not have the View Topology or Manage Topology permission, or this action is not allowed because one or more of the specified RASes are not in a suitable upgrade state.
404	Not found	One or more of the specified RAS UUIDs do not exist.

Body:

{"action": "PREPARE"}

OR

{"action": "START"}

Upload RAS Upgrade File

Request: POST /ras-upgrade-file

Description: Uploads the RAS upgrade file to Central. The uploaded file can later be used to upgrade RASes via Central. The file will be overwritten if it already exists.

Request body:

The RAS upgrade zip file, which must be uploaded according to the multipart/form-data standard (RFC 2388).

Response status codes:

Code	Meaning	Returned When
201	Created	Successful
400	Bad	The request is not a valid multipart/form-data upload, or the provided file is

	Request	invalid
403	Forbidden	The user does not have the Manage Topology permission

Response entity body:

- On success: No body.
- On failure: If the failure is due to a bad request, the failure reason will be returned in the following format:

{

"localizedMessage":"Invalid upgrade file. Make sure you are uploading a valid RAS upgrade zip.",

"details":"Required files are missing from the uploaded zip"

}

Get RAS Upgrade File Version

Request: GET /ras-upgrade-file/version

Description: Returns the version of the RAS upgrade file that has been uploaded to Central (see Upload RAS Upgrade File).

Response status codes:

Code	Meaning	Returned When
200	OK	Successful
403	Forbidden	The user does not have the View Topology or Manage Topology permission

Response entity body:

- On sucess: The version name in plain text, or empty body if no RAS upgrade is currently stored on Central.

Example:

10.70

Delete RAS upgrade file

Request: DELETE /ras-upgrade-file

Description: Deletes the RAS upgrade file.

Examples:

DELETE /ras-upgrade-file

Response status codes:

Code	Meaning	Returned When
200	OK	
403	Forbidden	Access denied user does not have the manage topology permission or when a RAS is consuming the upgrade file

Update a Specific Worker

Request: PUT /workers/{workerId}

Description: Update an existing worker.

Request path variables:

Attribute	Description
workerId	The ID of the worker to be updated.

Request entity body:

The body of this request must include a JSON object with the following format:

JSON for the worker

{

"groups":["worker_Operator_Path"

], "active":false

}

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Updated the default role successfully.
403	Forbidden	The user doesn't have Manage Topology permission.
404	Not found	The requested worker is not found.

Get All Workers

Request: GET /workers

Description: Retrieves all the workers.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested workers were found.
403	Forbidden	The user doesn't have View Topology or Manage Topology permissions.

Response entity body:

- on success: Returns a JSON object with the following format:

[

{

"uuid":"a97e30da-179e-4f19-af93-453c33338f53", "installPath":"c:/jenkins/workspace/carmel-demo-deployment/oo/central", "os":"Windows Server 2008",

"jvm":"1.7.0_13",

"description":"a97e30da-179e-4f19-af93-453c33338f53", "dotNetVersion":"4.x", "hostName":"VMCNCDEV41.devlab.ad",

"groups":[

"worker_Operator_Path"

], "active":true,

"status":"RUNNING"

},

{

"uuid":"4440c50e-79d1-45d2-a8dc-94bc42eb9b1f", "installPath":"c:\\jenkins\\workspace\\carmel-demo-deployment\\oo\\worker", "os":"Windows Server 2008",

"jvm":"1.7.0_13",

"description":"4440c50e-79d1-45d2-a8dc-94bc42eb9b1f", "dotNetVersion":"4.x", "hostName":"VMCNCDEV41.devlab.ad",

"groups":[

"Worker_Operator_Path"

], "active":false,

"status":"RUNNING"

}

]

See the following statuses for more information.

Worker Status

The following are the possible values of the status attribute, which appears in the APIs:

Status	Description
RUNNING	Normal status
IN_ RECOVERY	The worker is being recovered, but not finished. Once the recovery is finished it will change the status to RECOVERED
RECOVERED	Once the recovery is finished it will change the status to RECOVERED. If this worker startd again and performs a keepalive to the server it will change to RUNNING. If this worker has permanently stopped it will not change and remain RECOVERED.
FAILED	This is the initial status when the worker is created before keepalive is sent for the first time

Get All Workers Groups

Request: GET /workers-groups

Description: Return a list of Workers groups.

Response status code:

Code	Meaning	Returned When
200	OK	
403	Forbidden	The user does not have View Topology or Manage Topology permission.

Response entity body:

- on success: Returns a JSON array of the Workers Groups with the following format:

[

"RAS_Group_1", "RAS_Group_2", "RAS_Group_3"

]

Assign Workers to a Workers Group

Request: PUT /workers-groups/{name}/workers/{workersUuids}

Description: Assign Workers to a group.

Request path variables:

Attribute	Description
name	The name of the Workers group to add
WorkersUuids	The workersUuids of the Worker(s) to be added to the group

Response status codes:

Code	Meaning	Returned When
204	No Content	
400	Bad Request	
403	Forbidden	The user doesn't have Manage Topology permission.
404	Not Found	The requested worker is not found.

Remove Workers from a Workers Group

Request: DELETE /workers-groups/{name}/workers/{workersUuids}

Description: Remove Workers from a Workers Group.

Request path variables:

Attribute	Description
name	The name of the Workers Group to remove from
workersUuids	The uuid of the Worker(s) to remove from the group

Response status codes:

Code	Meaning	Returned When
204	No Content	
400	Bad Request	
403	Forbidden	The user does not have Manage Topology permission.
404	Not Found	The requested worker is not found or worker group does not exist.

Delete a Worker

Request: DELETE /workers/{workersUuid}

Description: Delete workers.

Request path variables:

Attribute	Description
workersUuid	List of worker uuids

Response status codes:

Code	Meaning	Returned When

204	OK	No content, workers deleted successfully.
400	Bad Request	When you try to delete a worker that is still running.
403	Forbidden	The user does not have Manage Topology permission.
404	Not Found	The requested worker is not found.

Firewall friendly APIs

Register Reverse RAS

Request: POST /reverse-rases

Description: Connect to and register reverse RAS.

Request Entity Body:

JSON object of worker object

Attribute	Type	Description	Comments
ipAddress	String	Hostname/IP of the reverse RAS	Mandatory
listenPort	Integer	The port on which the reverse RAS is listening	Mandatory
sharedSecret	String	Phrase used to authenticate the connection to the reverse RAS	Mandatory
protocolType	String	Protocol used to connect to the reverse RAS	Mandatory Possible values: ws – regular connection (http initiated) wss – secured connection (https initiated)

{

"ipAddress":"10.20.30.40",

"listenPort":8081,

"sharedSecret":" Admin111",

"protocolType":"wss"

}

Examples:

POST /reverse-rases

Response status codes:

Code	Meaning	Returned When
201	Created	
400	Bad Request	
403	Forbidden	Access denied user does not have manage topology permission
500	Internal Server Error	RAS creation failed. For more details see log files

Response entity body:

On success: Returns a JSON object of the new worker with its uuid.

{

"uuid":"d10d4382-ae54-49a1-9e90-2cdfeac4f949",

"ipAddress":"10.20.30.40",

"listenPort":1234,

"sharedSecret":"Admin111",

"protocolType":"wss"

}

Update Reverse RAS

Request: PUT /reverse-rases/{uuid}

Description: Update reverse RAS details.

Request path variables:

Attribute	Description
uuid	RAS unique identifier

Request Entity Body:

JSON object of worker object.

Attribute	Type	Description	Comments
ipAddress	String	Hostname/IP of reverse RAS	Mandatory
listenPort	Integer	Port reverse RAS listen on	Mandatory
sharedSecret	String	Phrase used to authenticate connection to reverse RAS	Mandatory
protocolType	String	Protocol used to connect to reverse RAS	Mandatory Possible values: ws – regular connection (http initiated) wss – secured connection (https initiated)

Examples:

PUT /reverse-rases/d10d4382-ae54-49a1-9e90-2cdfeac4f949

{

"ipAddress":"16.60.202.145",

"listenPort":8081,

"sharedSecret":"Abcabc12",

"protocolType":"ws"

}

Response status codes:

Code	Meaning	Returned When
200	OK	RAS was updated
400	Bad Request	Illegal arguments
403	Forbidden	Access denied user does not have manage topology permission
404	Not Found	RAS with given uuid doesn't exist
500	Internal Server Error	RAS update failed. For more details see log files

Response entity body:

On success: Returns a JSON object of the updated RAS

Test Connection to Reverse RAS

Request: POST /reverse-rases/test

Description:

Open connection to the worker and close it after success.

Request Entity Body:

Attribute	Type	Description	Comments
ipAddress	String	Hostname/IP of reverse RAS	Mandatory
listenPort	Integer	Port reverse RAS listen on	Mandatory
sharedSecret	String	Phrase used to authenticate connection to reverse RAS	Mandatory
protocolType	String	Protocol used to connect to reverse RAS	Mandatory Possible values:

			ws – regular connection (http initiated) wss – secured connection (https initiated)

JSON object of worker object

{

"ipAddress":"10.20.30.40",

"listenPort":1234,

"sharedSecret":"Admin111",

"protocolType":"wss"

}

Examples:

POST /reverse-rases/test

Response status codes:

Code	Meaning	Returned When
200	OK	Connection tested, regardless of success or failure
400	Bad Request	Illegal arguments
500	Internal Server Error	RAS test failed. For more details see log files

Response entity body:

On success: Returns test connection status code:

0 – test connection success

1 – failure, see logs (unable to identify specific error)

2 – timeout

3 – connection already opened to the requested destination (ip:port)

4 – Invalid configuration

5 – SSL certificate validation failure

6 – Wrong RAS connection (occurs when for already registered RAS we test /edit connection with providing ip:port of different RAS)

7 – Invalid credentials (in our case wrong shared secret)

8 – No Credentials provided (can be only theoretically)

9 – Connection refused (port is closed)

Get Reverse RAS Proxy Configuration

Request: GET /reverse-ras-proxy

Description: Retrieve proxy configuration for reverse RAS connection.

Response status codes:

Code	Meaning	Returned When
200	OK	Successful (empty body if not defined)
403	Forbidden	Access denied user does not have manage topology permission

Response entity body:

on success: Returns a JSON object with proxy configuration

{

"proxyAddress":"100.201.130.5",

"proxyListenPort":3322,

"proxyUserName":”admin”,

"proxyPassword":”admin”

}

Create Reverse RAS Proxy Configuration

Request: POST /reverse-ras-proxy

Description: Create a reverse RAS proxy configuration.

Attribute	Type	Description	Comments
proxyAddress	String	Hostname/IP of forward RAS	Mandatory
proxyListenPort	Integer	Port forward proxy listen on	Mandatory
proxyUserName	String	Forward proxy user	Optional
proxyPassword	String	Forward proxy password	Optional Required if proxy user provided

Examples:

POST /reverse-ras-proxy

Request Entity Body:

JSON object of proxy configuration object

{

"proxyAddress":"100.201.130.4",

"proxyListenPort":3322,

"proxyUserName":"admin",

"proxyPassword":"admin"

}

Response status codes:

Code	Meaning	Returned When
200	OK	Successful
400	Bad Request	Illegal arguments
403	Forbidden	Access denied user does not have manage topology permission

Response entity body:

on success: Returns a JSON object of the new proxy configuration

Update Reverse RAS Proxy Configuration

Request: PUT /reverse-ras-proxy

Description: Update reverse RAS proxy configuration.

Attribute	Type	Description	Comments
proxyAddress	String	Hostname/IP of forward RAS	Mandatory
proxyListenPort	Integer	Port forward proxy listen on	Mandatory
proxyUserName	String	Forward proxy user	Optional
proxyPassword	String	Forward proxy password	Optional Required if proxy user provided

Examples:

Request Entity Body:

JSON object of proxy configuration object

{

Develop

"proxyAddress":"100.201.130.5",

"proxyListenPort":3322,

"proxyUserName":"admin",

"proxyPassword":"admin"

}

Examples:

PUT /reverse-ras-proxy

Response status codes:

Code	Meaning	Returned When
200	OK	Successful
400	Bad Request	Illegal arguments
403	Forbidden	Access denied user does not have manage topology permission

Response entity body:

on success: Returns a JSON object with proxy configuration

Delete Reverse RAS Proxy Configuration

Request: DELETE /reverse-ras-proxy

Description:

Delete proxy configuration for reverse RAS connection.

Examples:

DELETE /reverse-ras-proxy

Response status codes:

Code	Meaning	Returned When
200	OK	
403	Forbidden	Access denied user does not have manage topology permission

Response entity body:

on success: No return value

Users

The Users API allows you to retrieve, update, create and delete users.

Create New Internal User

Request: POST /users

Description: Adds a new internal user.

Request entity body:

The body of this request must include a JSON object with the following format:

JSON for a user with a password and roles

{

"username":"mranderson",

"password":"12345", "roles":[

{"name":"EVERYBODY"},

{"name":"PROMOTER"}

]

}

If roles are provided with an empty array, the user is granted with the role that was set as the default.

Note: Do not use the me user name as this is reserved.

Response status codes:

Code	Meaning	Returned When
201	Successful (Created)	An internal user was created successfully.
400	Bad Request	When the wrong parameters are entered.
403	Forbidden	The user who executed this command does not have the Manage Security Configuration permission.
409	Conflict	The given username already exists.

Response entity body:

- on success: Returns a JSON object of the created Internal User with the following format:

{

"displayName":"mranderson", "userId":"mranderson", "emails":null,
"roles":[

"END_USER", "PROMOTER",

"SYSTEM_ADMIN"

], "permissions":null

}

Update Existing User

Request: PUT /users/{username}

Description: Update an existing internal user

Request path variables:

Attribute	Description
username	The name of the internal user to update.

Request entity body:

The body of this request must include a JSON object with the following format:

JSON for an Internal User update with both optional and mandatory fields:

{

"password": "12345",

"roles":[

{"name":"EVERYONE"},

{"name":"PROMOTER"}

],

"username": "mranderson"

}

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Updated the user successfully.
400	Bad Request	
403	Forbidden	The user who executed this command does not have the Manage Security Configuration permission.
404	Not Found	The user to update does not exist.
409	Conflict	Trying to rename a user to a name that already exists.

Response entity body:

- on success: Returns a JSON object of the updated internal user.

{

"displayName":"mranderson", "userId":"mranderson", "emails":null,
"roles":[

"END_USER", "PROMOTER",

"SYSTEM_ADMIN"

],

"permissions":null

}

Delete an Internal User

Request: DELETE /users/{userNames}

Description: Deletes users according to a specific list of user names.

Request path variables:

Attribute	Description	Required
userNames	The names of internal users to delete.	Yes

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The request was finish with success.
403	Forbidden	The user who executed this command does not have the Manage Security Configuration permission.

Note: A logged in user cannot delete their own internal user account. In this case the response status will be 200 OK and in the response body the result will be FORBIDDEN.

Response Body:

A map containing the user Ids as keys and the delete result as values. The values are predefined values: SUCCESS, FORBIDDEN, NOT_FOUND.

Example:

{

userName1: "FORBIDDEN",

userName2: "SUCCESS",

userName3: "NOT_FOUND"

}

Get Users

Request: GET /users?domain=internal

Description: Retrieves users

Request parameters:

Attribute	Description	Default Value	Required
domain	The location from which the user's provider should be retrieved. Internal stands for internal users.	No	Yes

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Returned the requested users list.
403	Forbidden	The user does not have View Security Configuration or Manage Security Configuration permission.

Response entity body:

- on success: Returns a JSON object with the following format:

[

{

"displayName":"admin", "userId":"admin", "emails":null,
"roles":[

"ADMIN"

], "permissions":null

},

{

"displayName":"mranderson", "userId":"mranderson", "emails":null,
"roles":[

"END_USER"

], "permissions":null

},

{

"displayName":"rothjohn", "userId":"rothjohn", "emails":null,
"roles":[

"EVERYONE"

], "permissions":null

}

]

Get Session's User

Request: GET /users/me

Description: Retrieves this session's user.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The session's user was returned.

Response entity body:

- on success: Returns a JSON object with the following format:

{

"displayName":"admin", "userId":"admin", "emails":null,
"roles":[

"ADMIN"

], "permissions":[

"cpManage", "cpRead", "topologyManage", "flowPermissionManage", "topologyRead",
"securityConfigManage", "securityConfigRead", "systemSettingsRead", "systemSettingsManage",
"scheduleManage", "scheduleRead",

"configurationItemManage", "configurationItemRead",

"othersRunsManage"

]

}

LW SSO

The LW SSO API allows you to configure LW SSO.

Get LW SSO Configuration

Request: GET /authns/lwsso-config

Description: Retrieves the lightweight SSO configuration.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The LWSSO configuration was returned.
403	Forbidden	The user does not have View Security Configuration or Manage Security Configuration permission.

Response entity body:

- on success: Returns a JSON object with the following format:

{ "enabled":false,

"initString":"**********", "domain":"mydomainnew1.com", "protectedDomains":

[

"mydomainnew1.com",

"mydomainnew2.com"

]

}

Update LW SSO configuration

Request: PUT /authns/lwsso-config

Description: Updates the lightweight SSO configuration.

Request entity body:

The body of this request must include a JSON object with the following format (some properties are optional):

{

"enabled":false,

"initString":"myInitString",

"domain":"mydomainnew1.com",

"protectedDomains":

[

"mydomainnew1.com",

"mydomainnew2.com"

]

}

Note: The initString must be at least 12 characters long.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Updated the LW SSO configurations successfully.
400	Bad Request	
403	Forbidden	The user does not have Manage Security Configuration permission.

Response entity body:

- on success: Returns a JSON object of the saved configurations.

{

"enabled":false,

"initString":"**********",

"domain":"mydomainnew1.com",

"protectedDomains":

[

"mydomainnew1.com",

"mydomainnew2.com"

]

}

Authentication

The Authentication API allows to enable and disable user authentication.

Get Authentication Configurations

Request: GET /authns

Description: Retrieves the authentication status

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The authentication status we returned

Response entity body:

{

"enable":true,

"domains":[

"Internal"

]

}

Update Authentication Configurations

Request: PUT /authns

Description: Updates the authentication configurations.

Request body:

{ "enable":true

}

Response status codes:

Code	Meaning	Returned When
204	Successful (no-content)	The authentication configuration were updated.
403	Forbidden	The user does not have Manage Security Configuration permission.
404	Not Found	The specified path was not found.

Roles

The Roles API allows you to configure roles.

Get Specified Role

Request: GET /roles/{roleName}

Description: Retrieves a role according to the specified role name.

Request path variables:

Attribute	Description
roleName	The name of the required role.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested role was found.
403	Forbidden	The user does not have View Security Configuration or Manage Security Configuration permission.
404	Not Found	The requested role was not found.

Response entity body:

- on success: Returns a JSON object with the following format:

{

"name":"ADMINISTRATOR",

"description":"Administration Role",

"permissions":[

"flowPermissionManage",

"cpManage",

"dashboardRead",

"configurationItemRead",

"cpRead",

"configurationItemManage",

"scheduleManage",

"systemSettingsManage",

"scheduleRead",

"securityConfigRead",

"topologyRead",

"flowDebug",

"securityConfigManage",

"topologyManage",

"systemSettingsRead",

"othersRunsManage"

],

"groupsNames":[

]

}

groupsNames attribute refers to the LDAP groups mapping. An empty array indicates that there is no mapping to any LDAP group.

Get All Roles

Request: GET /roles

Description: Retrieves all the existing roles.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested roles were found.
403	Forbidden	The user doesn't have View Security Configuration or Manage Security Configuration permission.

Response entity body:

- on success: Returns a JSON object with the following format:

[

{

"name":"ADMINISTRATOR", "permissions":[

"securityConfigRead", "cpRead", "topologyManage", "securityConfigManage", "configurationItemRead", "scheduleManage", "topologyRead", "othersRunsManage",

"configurationItemManage", "systemSettingsManage", "flowPermissionManage", "cpManage",

"scheduleRead",

"systemSettingsRead"

],

"groupsNames":[],

"description":"Administration Role"

},

{

"name":"EVERYBODY",

"permissions":[],

Create New Role

Request: POST /roles

Description: Adds a new role Request entity body:

The body of this request must include a JSON object with the following format:

JSON for a role configuration with both optional and mandatory fields:

{ "name":"NEW_ROLE",
"permissions":["cpRead",

"cpManage"

],

"groupsNames":[],

"description":"New Role"

}

description and groupsNames are optional.

The groupsNames refers to the LDAP groups that should be mapped to this role.

Response status codes:

Code	Meaning	Returned When
201	Successful (Created)	A new role was created.
400	Bad Request	The data in the body is incorrect.
403	Forbidden	The user does not have Manage Security Configuration permission.
409	Conflict	When the user tries to add a role which is already exists.

Response entity body:

- on success: Returns a JSON object of the created role with the following format:

{

"name":"Super Power",

"description":"An all permissions role!", "permissions":[

"flowPermissionManage", "cpManage", "dashboardRead", "configurationItemRead", "cpRead",

"configurationItemManage", "scheduleManage", "systemSettingsManage", "scheduleRead", "securityConfigRead", "topologyRead", "flowDebug", "securityConfigManage", "topologyManage", "systemSettingsRead", "othersRunsManage"

], "groupsNames":[

"Super Group"

]

}

Update an Existing Role

Request: PUT /roles/{roleName}

Description: Update an existing role

Request path variables:

Attribute	Description
roleName	The name of the role to update.

Request entity body:

The body of this request must include a JSON object with the following format:

JSON for a Role update with both optional and mandatory fields:

{

"description":"Not super power anymore", "groupsNames":[

"Not Super Group"

],

"name":"Not Super Power", "permissions":[

"othersRunsManage", "flowPermissionManage", "securityConfigRead",

"securityConfigManage"

]
}

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Updated the role successfully.
400	Bad Request	The JSON body is incorrect.
403	Forbidden	The user does not have Manage Security Configuration permission.
404	Not Found	When the role does not exist.

Response entity body:

- on success: Returns a JSON object of the updated internal user.

{

"description":"Not super power anymore", "groupsNames":[
"Not Super Group"

],

"name":"Not Super Power", "permissions":[

"othersRunsManage",
"flowPermissionManage",
"securityConfigRead",

"securityConfigManage"

]

}

Delete a Role

Request: DELETE /roles/{roleName}

Description: Deletes a role according to the specified role name.

Request path variables:

Attribute	Description
roleName	The identifier of the role name to delete.

Response status codes:

Code	Meaning	Returned When
204	Successful (no-content)	The role was deleted successfully.
403	Forbidden	The user does not have Manage Security Configuration permission.
404	Not found	

Get the Default Role

Request: GET /roles/default-name

Description: Retrieves the default role.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The default role found.
403	Forbidden	The user does not have View Security Configuration or Manage Security Configuration permission.

Response entity body:

- on success: Returns a JSON object with the following format:

{"defaultRole":"EVERYBODY"}

defaultRole attribute maps between the default role and an existing one.

Upload the Deployment Process File

Request: POST /deployments/{deploymentProcessId}/files

Description: Uploads one or more content pack files, and associates them with an existing deployment process (see Create Deployments). The request should be formatted according to the "multipart/form-data" standard for uploading files to a web server (RFC 2388).

Note: The request body should be the contents of the content pack files to be deployed.

Request path variables:

Attribute	Description
deploymentProcessId	The deployment process ID to associate the files with. The ID is returned when creating a new deployment process.

Example:

POST /deployments/109600004/files

Response status codes:

Code	Meaning	Returned When
200	OK	The files were uploaded and associated with the specified deployment process.
400	Bad Request	Some or all of the uploaded files have failed basic validation.
403	Forbidden	The user does not have Manage Content Packs permission.
404	Not Found	The deployment process object with the specific id does not exist.
409	Conflict	Some or all of the uploaded content packs have already been added to this deployment process.
500	Internal Server Error	The HTTP request was not a valid "multipart/form-data" request (RFC 2388).

JSON response example:

The response body includes details for each of the uploaded files. The signDetails property contains the content pack's digital signature details. The deleteUrl and deleteType properties represent a REST request, which you can send to remove the particular file from this deployment process.

{

"files":

[

{

"name":"cp-signed.jar",

"size":10095,

"fileUploadId":110800009,

"deleteUrl":"/deployments/109600005/files/110800009",

"deleteType":"DELETE",

"signDetails":

{"signStatus":"signed",

"signedBy":"CN=qa-MAINDC-CA, DC=qa, DC=ad, DC=com",

"warnings":[],

"certs":

[

{

"certType":"X.509",

"certDn":"CN=Administrator, CN=Users, DC=qa, DC=ad, DC=com",

"certKeystoreEntryAlias":null,

"validityTimeFrom":1417599979000,

"validityTimeNotAfter":1480758979000,

"validityTimeNotBefore":0,

"supportCodeSign":false

},

{

"certType":"X.509",

"certDn":"CN=qa-MAINDC-CA, DC=qa, DC=ad, DC=com",

"certKeystoreEntryAlias":null,

"validityTimeFrom":1390826569000,

"validityTimeNotAfter":1706359968000,

"validityTimeNotBefore":0,

"supportCodeSign":false

}

],

"trusted":false}

}

]

Get Entitlements Per Path and Roles

Request: GET /roles/{rolesNames}/entitlements/{path}

Description: Retrieves the entitlements for the requested roles and path.

Example:

GET /roles/ADMINISTRATOR,EVERYBODY,PROMOTER/entitlements/Library/Flows/flow.xml

Request path variables:

Attribute	Description
rolesNames	The roles for which the entitlements are requested.
Path	The full path of the resource, including .xml. Note: The "/" in the path must not be encoded.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested entitlements were found.
400	Bad request	
403	Forbidden	The user does not have Manage Content Permissions permission.
404	Not Found	The requested path was not found.

Response entity body:

- on success: Returns a JSON object with the following format:

{

"ADMINISTRATOR":["RUN",

"VIEW"

], "EVERYBODY":[

Develop

], "END_USER":[

], "SYSTEM_ADMIN":[

], "PROMOTER":[

"RUN",

"VIEW"

]

}

Note: Existing entitlements are:

- RUN – run flow,
- VIEW – view flow,
- VIEW_EXECUTE – view and use system account.

Update Path Entitlement Per Role

Request: PUT /roles/{roleName}/entitlements/{path}

Description: Update the entitlements of the requested role and path. The path can be both a flow and configuration item. The privileges for flows are RUN and VIEW, and for configuration items, VIEW_ EXECUTE.

Example:

PUT /roles/SYSTEM_ADMIN/entitlements/Library/cp-parallel

Request path variables:

Attribute	Description
roleName	The role for which the entitlements are impact.
Path	The full path of the resource, including .xml. Note: the "/" in the path must not be encoded.

Request entity body:

The body of this request must include a JSON object with the following format:

JSON for an entitlement path update with both optional and mandatory fields:

{

"privileges":["VIEW", "RUN"

], "isRecursive":true

}

Note: The default value for isRecursive is false.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Updated the role successfully.
400	Bad Request	The request contains the wrong data, for example, no existing privileges.
403	Forbidden	The user does not have Manage Content Permissions permision.
404	Not found	The specified path was not found.

Response entity body:

- on success: Returns a JSON object of the updated path entitlement on the role.

{

"privileges":["VIEW", "RUN"

], "isRecursive":true

}

System Information and Settings

Create a System Configuration Item

Request: POST /config

Description: Creates a system configuration item.

Request entity body:

The body of this request needs to include a JSON object with the following format:

{

"value":"value",

"key":"my.test.key"

}

Response status codes:

Code	Meaning	Returned When
201	Created	A system configuration item was created successfully.
403	Forbidden	The user does not have Manage System Settings permission.
409	Conflict	The system configuration item is already exists.

Response entity body:

- on success: Returns a JSON object of the created system configuration item with the following format:

{

"id": "1179648", "key":"myKey", "value":"value"

}

In addition, a location header containing a URI to retrieve the created system configuration item:

/config/myKey

Get All System Configuration Items

Request: GET /config

Description: Retrieves all system configuration items.

Response status codes:

Code	Meaning	Returned When
200	OK	All existing system configuration items were retrieved.
403	Forbidden	The user does not have View System Settings or Manage System Settings permission.

Response entity body:

- on success: Returns a JSON object with the following format:

{

"key 1":"value 1", "key 2":" value 2",

...

"key n":" value n"

}

Get a System Configuration Item

Request: GET /config/{key}

Description: Retrieves a system configuration item by key.

Request path variables:

Attribute	Description
key	The key of the requested system configuration item.

Response status codes:

Code	Meaning	Returned When
200	OK	The requested system configuration item was retrieved successfully.
403	Forbidden	The user does not have View System Settings or Manage System Settings permission.
404	Not Found	No system configuration was retrieved.

Response entity body:

- on success: Returns a JSON string with the value of the requested system configuration item.

Update System Configuration Item

Request: PUT /config/{key}

Description: Updates an existing system configuration item.

Request path variables:

Attribute	Description
key	The key of the system configuration item to update.

Request entity body:

The body of this request needs to include the new value of the system configuration item, as plain text (no JSON encoding).

Response status codes:

Code	Meaning	Returned When
202	Accepted	The system configuration item was updated successfully.
403	Forbidden	The user does not have Manage System Settings permission.
404	Not Found	The requested system configuration item was not found.

Response entity body:

- on success: Returns a JSON value of the updated configuration item's ID.

Get Database Usage Statistics

Request: GET /db-statistics

Description: Retrieves the HPE OO database usage statistics (in MB) according to the given request parameters.

Note: Database usage represents the sum of database segments/pages in use by HPE OO. It does not reflect the overall “size-on-disk” and/or database transaction log.

Request parameters:

Attribute	Type	Description	Required	Default
months	List of numbers	The months to retrieve the statistics for. An empty list means all months.	No	false
years	List of numbers	The years to retrieve the statistics for. An empty list means all years.	No	false

Example:

GET/db-statistics?months=1,2,3&years=2015

Response status codes:

Code	Meaning	Returned When
200	OK	The requested database usage statistics was returned.
403	Forbidden	The user who executed this command does not have permission for Manage cleanup data or to view or manage system settings.

Response entity body:

An object containing two fields: lastUpdate, which represents the last time the database size was checked, and data, which is a map containing all of the database statistics that match the request.

The first level of the map is the statistics years. Each year is mapped to the different months, and each month is mapped to the different days. Each day contains the statistics element. This statistics element contains the total usage of the database, the debugger usage, the execution usage and the audit usage for that time (in MB).

Example:


```
{
 "lastUpdate":1182139200000,
 "data":
  {
   "2014":{
    "12":{
     "31": {
      "totalUsage": 3,
      "executionUsage": 0,
      "debuggerUsage": 0,
      "auditUsage": 0
     }
    }
   },
   "2015":{
    "1":{
     "1": {
      "totalUsage": 1050,
      "executionUsage": 568,
      "debuggerUsage": 150,
      "auditUsage": 0
     },
     "2": {
      "totalUsage": 3,
      "executionUsage": 756,
```

Develop


```
        "debuggerUsage": 150,
        "auditUsage": 3
       }
      },
     }
    }
  }
}
```


Get System Log Level

Request: GET /loglevel

Description: Retrieves the system default run log level (persistence level).

Example: GET/loglevel

Response status codes:

Code	Meaning	Returned When
200	OK	The requested log level was returned.
403	Forbidden	The user does not have the permission View System Settings or Manage System Settings

Response entity body:

- On success: Returns a quoted JSON string of the system log level.

Possible values: STANDARD, EXTENDED

Example:

"STANDARD"

Update System Log Level

Request: PUT /loglevel

Description: Updates the system default run log level (persistence level).

Example: PUT /loglevel

Request entity body:

The body of this request must include one of the predefined values (quoted JSON string): STANDARD, EXTENDED.

Example:

"STANDARD"

Response status codes:

Code	Meaning	Returned When
204	No Content	The requested log level was updated successfully.
400	Bad Request	The provided log level does not match the predefined values (Case sensitive).
403	Forbidden	The user does not have the permission Manage System Settings

Install License from file

Request: PUT /licensing/file

Description: Install and replace the current license of OO (file).

Expected header: Content-Type : multipart/form-data

Response status codes:

Code	Meaning	Returned When
200	OK	The license successfully installed.
400	Bad Request	The user provided an invalid license file.
403	Forbidden	User does not have the system manage permission.

Install License from key

Request: PUT /licensing/key

Description: Install and replace the current license of OO (license key).

Expected header: Content-Type : application/json

Response status codes:

Code	Meaning	Returned When
200	OK	The license successfully installed.
400	Bad Request	The user provided an invalid license file.
403	Forbidden	User does not have the system manage permission.

Get General Setting by Key

Request: GET /general-settings/{key}

Description: Returns the value of the requested general setting. The following keys are allowed:

- masthead.extra.title
- roi.description
- system.level.flow.timeout

Example:

GET /general-settings/system.level.flow.timeout

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Request is successful and the value is returned.
404	Not Found	Requested key is not a general settings key and cannot be found.

Get HPE OO Version

Request: GET /version

Description: Retrieves information about the HPE OO version.

Response entity body:

- on success: Returns a JSON object with the HPE OO version information:

{

version: "10.20",

revision: "61583",

build ID: "2014-11-02_15-03-32",

build number: "8",

build job name: "branch-product
}

System Notifications

Get System Notifications

Request: GET /notifications

Description: Retrieves the available notifications, their description and status

Response status codes:

Code	Meaning	Returned When
200	OK	The requested notifications were returned.
402	Forbidden	License required to use premium feature.
403	Forbidden	The user does not have the permission to View System Settings or Manage System Settings.

Response entity body

{

```
"name": "authenticationDisabled",
```

Develop


```
"mailEnabled": false,
"description": "Central Authentication was disabled",
"properties":{ }
},
{
"name": "runningTimeExceeded",
"mailEnabled": false,
"description": "Flows running for more than # hours",
"properties":{
" hours": "24"
}
```


Set System Notifications

Request: PUT/notifications

Description: Sets the details and status fort the system notifications.

Response status codes:

Code	Meaning	Returned When
200	OK	The requested notifications were returned.
402	Forbidden	License required to use premium feature.
403	Forbidden	The user does not have the permission to View System Settings or Manage System Settings.

Example


```
{
{
"name": "authenticationDisabled",
"mailEnabled": true,
"properties":{ }
},
{
"name": "runningTimeExceeded",
```

```
"mailEnabled": true,
"properties":{
" hours": "72"
}
}
}
```

> The request can only change the enabled status and properties. If a value is not provided, the previous value is retained.

Set System Notifications

Request: PUT/notification-channel/mail

Description: Sets the notification channel details

Response status codes:

Code	Meaning	Returned When
200	OK	The requested notifications were returned.
402	Forbidden	License required to use premium feature.
403	Forbidden	The user does not have the permission to View System Settings or Manage System Settings.

Example

```
{
"host": "smtpServer",
"port": "25",
username": "admin",
"password": "password",
"from": "senderAddress",
"to": "receiver1Address;receiver2Address"
}
```

> The PUT request overwrites the existing configuration. Any field that is left blank will not retain its previous value.

Get System Notifications Channel

Request: GET /notification-channel/mail

Description: Gets the notification channel details

Response status codes:

Code	Meaning	Returned When
200	OK	The requested notifications were returned.
402	Forbidden	License required to use premium feature.
403	Forbidden	The user does not have the permission to View System Settings or Manage System Settings.

Response entity body

```
{
"host": "smtpServer",
"port": "25",
username": "admin",
"password": "password",
"from": "senderAddress",
"to": "receiver1Address;receiver2Address"
}
```

Developing Actions

This topic shows you how to create @Actions as a Maven plugins to be used in OO.

Introduction

This topic provides content developers with guidelines for developing new operations for the Operations Orchestration platform. Operations are used to create new flows that can be executed inside HPE Operations Orchestration flow execution engine.

Introducing new content requires building an extension and deploying it to HPE OO Central or Studio. An extension is an artifact that contains actions written in Java or.NET.

Actions are code blocks used to create new operations for HPE Operations Orchestration. You can also use actions to create new operations for the CloudSlang platform. HPE Operations Orchestration can also execute operations and flows written in CloudSlang language.

System Requirements

- Java 1.7
- NET Framework 4.0 (required only if you want to create .NET actions)
- Maven 3.3.9 (if you want to build operations for CloudSlang)
- Maven 3.2.1
- Maven 3.0.5 (if you want to build an operation for Operations Orchestration 10.20 or lower versions)
- Java IDE (optionally, you may want to install a Java IDE with Maven support, like Eclipse or IntelliJ)
- Python IDE (optionally, you may want to install a Python IDE like PyCharm or Eclipse)

Prerequisites

> Java or .NET knowledge is required for developing new actions.

The action development process relies on Maven for resolving dependencies and building the project.

In order to properly comprehend all the topics discussed in this document, it is recommended that you read the “Maven Getting Started Guide” reference at the end of this document before continuing.

To develop CloudSlang content, it is recommended that you first read the “CloudSlang Tutorial” reference at the end of this document.

HPE OO Content Development

HPE OO Plugins

To create new content for HPE Operations Orchestration 10.x, you need to develop an extension called a plugin.

An OO plugin is a JAR file, packaged as a Maven plugin that contains one or multiple actions.

> Each plugin defines its own isolated classpath. Classpath isolation ensures that different plugins can use conflicting dependencies. For example, plugin A can use dependency X version 1.0 and plugin B can use the same X dependency, but with version 2.0. Classpath isolation ensures that you can use both plugins in the same flow regardless of the conflicting classpath issue.

HPE Operations Orchestration 10.x provides a simple way for creating Java actions by introducing the @Action annotation. See "Developing Actions" for more information.

HPE Operation Orchestration can also execute .NET actions. Actions written in .NET are referenced by a wrapping Java plugin. See ".NET Extensions" for more information.

> In previous versions of HPE Operations Orchestration, extensions were called IActions, They implemented the IAction interface. The IAction interface is now deprecated. Users writing new content should refrain from implementing the IAction interface and instead use the @Action annotation.

Developing Plugins for Java Actions

This section describes how to develop plugins containing only java actions, marked with the @Action annotation.

Using the Maven archetype **com.hp.oo.sdk:oo-plugin-archetype** you can create a skeleton for a plugin and a Studio project.

Preparing to Create a Plugin Using a Maven Archetype

Install Maven

Install Maven on your computer, adding the bin folder to the "Path" system variable. This enables you to run the `mvn` command from anywhere in the file system.

For more information on the Maven Installation process, check the “Installing Apache Maven” reference in the References section.

HPE Operations Orchestration SDK 10.6x depends on Maven 3.2.1. This version of Maven has a defect concerning the fact that it tries to resolve dependencies using external repositories even though we specify our internal Nexus artifact management system in the **settings.xml** file. The workaround is to force Maven to use a single repository – the internal repository. For more information, see the “Mirror Remote Maven Repositories” section.

Mirror Remote Maven Repositories

You can force Maven to use a single remote repository by mirroring all requests to remote repositories through an internal (company) repository. The internal repository must either contain all the desired artifacts, or it should be able to proxy the requests to other repositories. This setting is most useful when using an internal company repository with the Maven Repository Manager to proxy external requests.

To achieve this, configure a mirror for everything (*) in your **settings.xml** file. For complete guidelines, check the “Using Mirrors for Repositories” reference at the end of this document.

```
<settings>
  ...
  <mirrors>
    <mirror>
       <id>internal-repository</id>
       <name>Maven Repository Manager running on repo.mycompany.com</name>
       <url>http://repo.mycompany.com/proxy</url>
       <mirrorOf>*</mirrorOf>
    </mirror>
  </mirrors>
  ...
 </settings>
```

Create a local Maven repository

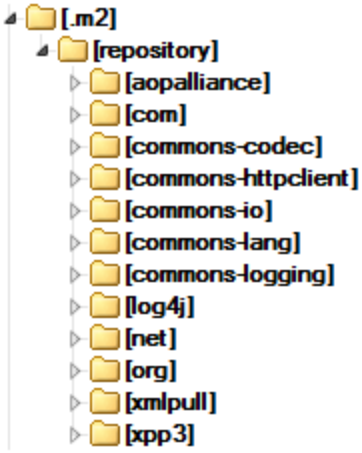
- Expand `sdk-dotnet-<version>.zip and sdk-java-<version>`.zip to:

  Windows: %HOMEPATH%\.m2\repository.

  Linux: $HOME/.m2/repository.

  > These files are located on the HPE OO ZIP file in the SDK folder.

Following is an example of a directory structure, if the files were correctly extracted:



Register the plugin archetype

- Open the command prompt and enter the following command:

  ```
  mvn archetype:crawl
  ```

  This updates the Maven archetype catalog under $HOME/.m2/repository.

Creating a Plugin Using a Maven Archetype

Create a sample project

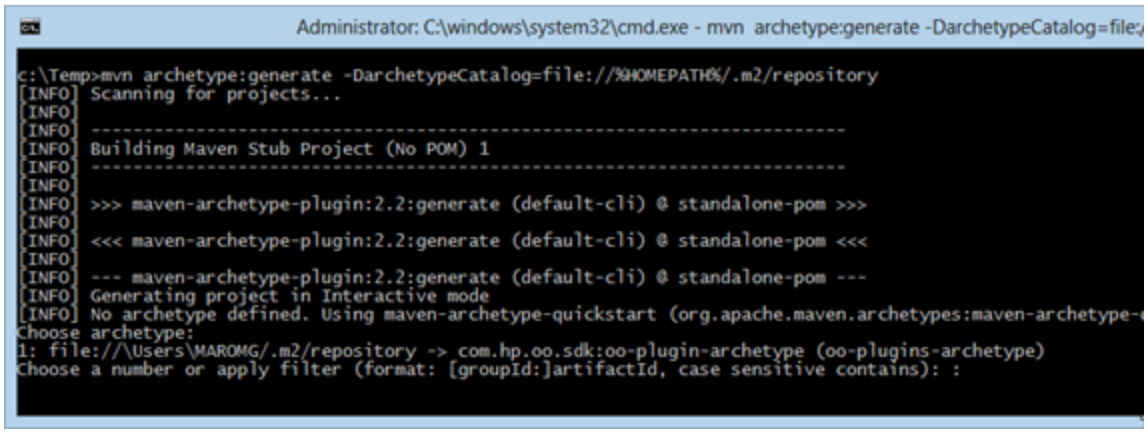
1. Go to the path where you want to create a sample plugin project, and enter the following command in the command line:

   ```
   mvn archetype:generate -DarchetypeCatalog=file://$HOME/.m2/repository
   ```

   Note: For Windows, use %HOMEPATH%.

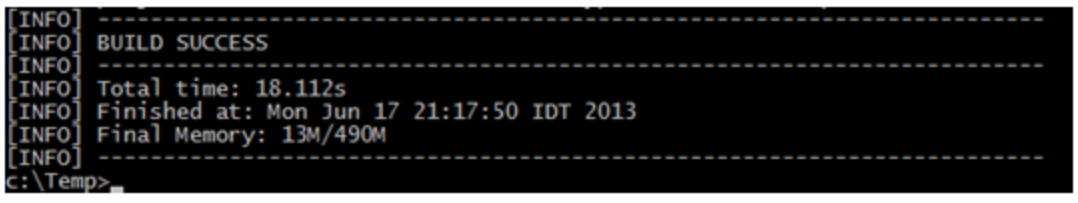
   This initiates the project creation. A list of archetypes found in the catalog appears. Select the number representing the **archetype com.hp.oo.sdk:oo-plugin-archetype**.

   In the example below, select 1.



2. During the archetype creation, enter the following details:
   - `groupId`: The group id for the resulting Maven project. `acmeGroup` is used in the example below.
   - `artifactId`: The artifact id for the resulting Maven project. `acmeArtifact` is used in the example below.
   - `package`: The package for the files in the project. The default for this option is the same as the `groupId`.
   - `uuid`: The UUID of the generated project. A randomly generated UUID is used in the example below.

The build finishes and a project is created.



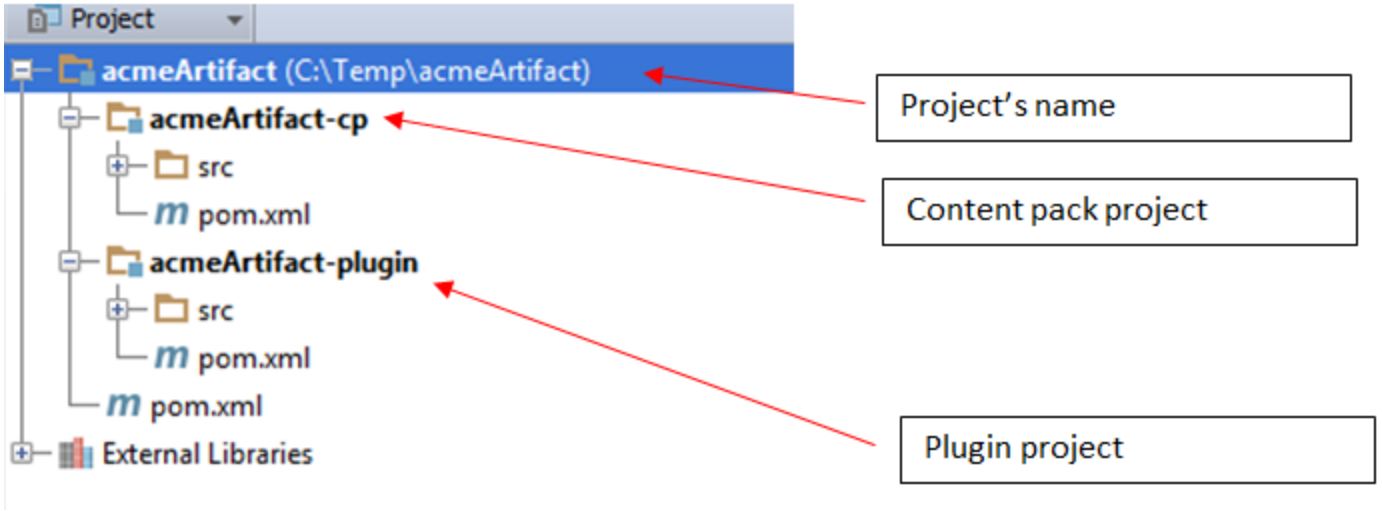
Open the Project in a Java IDE

The previous step created a new Java project with a Maven-based model.

Open this project in a Java IDE application.

The project contains two modules that have the same prefix as the provided artifact ID. One of the projects is a content pack project and the other is a plugin project.

For example:



Parent project

In the illustrated example, the parent project is called **acmeArtifact**.

By default it contains two modules—one a content pack and the other a plugin. This project groups @Actions and their relevant operations and flows into a single content pack.

> In large projects you can group actions in plugins based on functionality, protocols or technologies such as: ssh-plugin, http-plugin etc. When you create operations from the actions of a project, place them all in a single content pack project, disregarding the plugin in which it is contained.

Example: If you were developing a Microsoft Office integration, you might create several plugin projects—one for each Office version. But there would be a single content pack project containing the operations and flows.

Plugin project

In the example, the plugin project is called **acmeArtifact-plugin**.

This module contains the @Actions. When you are building this project (with Maven), the code inside is compiled and the resulting JAR file can be opened in Studio, and operations can be created from the @Actions inside.

Inside this module, a sample @Action can be found. You can delete it and write your own.

Content pack project

In the example, the content pack project is called **acmeArtifact-cp**.

This module represents the content pack. It includes any plugin modules upon which it is dependent, for example, **acmeArtifact-plugin** and its dependencies), as well as any flows, operations, and configuration items defined within it.

Creating a New Action

A Java action is a method that conforms to the signature `public Map<String, String> doSomething(parameters)`, annotated with the **@Action** annotation.

You can find the complete specification of the **@Action** annotation in the Developing Actions section.

Build Maven Plugin

To build your maven project, navigate to the location of the parent project, for example, c:\temp\acmeArtifact), open the command line and `run mvn clean install`.

You can see the artifacts resulted from the build process under the **target** folder of each module.

These artifacts are also installed in the local maven repository located at **%HOMEPATH%\.m2\repository or $HOME/.m2/repository**.

For example, you can find the resulted artifact from building **acmeArtifact-plugin** loacted in: **%HOMEPATH%\.m2\repository\acmeGroup\acmeArtifact-plugin\1.0.0\acmeArtifact-plugin-1.0.0.jar**.

Developing Plugins for .NET Actions

In order to create content using .NET actions, you need to:

1. Create a DLL file containing the implementation of the desired actions. The action class should implement an IAction interface.

2. Deploy the created DLL, including referenced libraries, to the local Maven repository, using `mvn install:install-file`. For more information on installing artifacts that were not built by Maven, see http://maven.apache.org/plugins/maven-install-plugin/usage.html.
3. Generate an HPE OO Maven plugin, wrapping the .NET action. To do this, you need to:
   a. Create a **pom.xml** file. For POM references, see http://maven.apache.org/pom.html.
   b. Under the <dependencies> tag, add a list containing all the required DLLs. Define all DLL artifacts using <type>dll</type>.
   c. Run the mvn install command from the folder containing the **pom.xml** file. This is assuming that the Maven bin folder is contained in the system path.

The result is the Maven plugin, placed in the target folder and installed to the local Maven repository. The target folder location is relative to the current folder.

The template of the **pom.xml** is:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

     <groupId>[my plugin groupId]</groupId>
     <artifactId>[my plugin artifactId]</artifactId>
     <version>[my plugin version]</version>

     <packaging>maven-plugin</packaging>

     <properties>
<oo-sdk.version>[THE LATEST HPE OO_SDK VERSION]</oo-sdk.version>
<oo-dotnet.version>[THE LATEST HPE OO_DOTNET VERSION]</oo-dotnet.version>
     </properties>

     <dependencies>
        <!-- required dependencies -->
        <dependency>
              <groupId>com.hp.oo.sdk</groupId>
              <artifactId>oo-dotnet-action-plugin</artifactId>
              <version>${oo-sdk.version}</version>
        </dependency>

        <dependency>
              <groupId>com.hp.oo.dotnet</groupId>
              <artifactId>oo-dotnet-legacy-plugin</artifactId>
```

```
            <version>${oo-dotnet.version}</version>
            <type>dll</type>
        </dependency>

        <dependency>
            <groupId>${project.groupId}</groupId>
            <artifactId>IAction</artifactId>
            <version>9.0</version>
            <type>dll</type>
        </dependency>
        <!-- end of required dependencies -->

        <dependency>
            <groupId>[groupId-1]</groupId>
            <artifactId>[artifactId-1]</artifactId>
            <version>[version-1]</version>
            <type>dll</type>
        </dependency>

        <dependency>
            <groupId>[groupId-2]</groupId>
            <artifactId>[artifactId-2]</artifactId>
            <version>[version-2]</version>
            <type>dll</type>
        </dependency>

       ...

        <dependency>
            <groupId>[groupId-n]</groupId>
            <artifactId>[artifactId-n]</artifactId>
            <version>[version-n]</version>
            <type>dll</type>
        </dependency>
    </dependencies>

    <build>
       <plugins>
          <plugin>
                <groupId>com.hp.oo.sdk</groupId>
                <artifactId>oo-action-plugin-maven-plugin</artifactId>
                <version>${oo-sdk.version}</version>
                <executions>
                    <execution>
                        <id>generate plugin</id>
                        <phase>process-sources</phase>
                        <goals>
                           <goal>generate-dotnet-plugin</goal>
                        </goals>
```

```
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Example: In the following example:

- The POM file is named **example.pom.xml**.
- The **my-dotnet-actions.dll** contains the desired actions.
- The generated Maven plugin is **com.example:my-dotnet-plugin:1.0**.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                       http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>my-dotnet-plugin</artifactId>
    <version>1.0</version>

    <packaging>maven-plugin</packaging>

    <properties>
       <oo-sdk.version>[THE LATEST HPE OO_SDK VERSION]</oo-sdk.version>
       <oo-dotnet.version>[THE LATEST HPE OO_DOTNET VERSION]</oo-dotnet.version>
    </properties>

    <dependencies>
        <!-- required dependencies -->
        <dependency>
            <groupId>com.hp.oo.sdk</groupId>
            <artifactId>oo-dotnet-action-plugin</artifactId>
            <version>${oo-sdk.version}</version>
        </dependency>
        <dependency>
            <groupId>com.hp.oo.dotnet</groupId>
            <artifactId>oo-dotnet-legacy-plugin</artifactId>
            <version>${oo-dotnet.version}</version>
            <type>dll</type>
        </dependency>
        <dependency>
            <groupId>${project.groupId}</groupId>
            <artifactId>IAction</artifactId>
```

```
            <version>9.0</version>
            <type>dll</type>
        </dependency>
        <!-- end of required dependencies -->

        <dependency>
            <groupId>com.example</groupId>
            <artifactId>my-dotnet-actions</artifactId>
            <version>1.0</version>
            <type>dll</type>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>com.hp.oo.sdk</groupId>
                <artifactId>oo-action-plugin-maven-plugin</artifactId>
                <version>${oo-sdk.version}</version>
                <executions>
                   <execution>
                      <id>generate plugin</id>
                      <phase>process-sources</phase>
                      <goals>
                         <goal>generate-dotnet-plugin</goal>
                      </goals>
                   </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Developing Plugins for Legacy Actions

> This is a deprecated way of writing actions. It is recommended to use the @Action annotation for developing new Java actions.

In order to create content using legacy actions, you need to:

1. Verify that you have a JAR containing the implementation of the desired actions, just like in version 9.x. The action class should implement an IAction interface.
2. Deploy the JAR, including referenced libraries, to the local Maven repository, using mvn install:install-file. For more information on installing artifacts that were not built by Maven, see

http://maven.apache.org/plugins/maven-install-plugin/usage.html.

3. Generate an HPE OO Maven plugin, wrapping the legacy actions library. To do this, you need to:
   a. Create a **pom.xml** file. For POM references, see http://maven.apache.org/pom.html.
   b. Run the `mvn install` command from the folder containing the **pom.xml** file. This is considering that the Maven bin folder is contained in the system path.

The result is the Maven plugin, placed in the target folder and installed to the local Maven repository. The target folder location is relative to the current folder.

The content of the **pom.xml** is:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                       http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>[my plugin groupId]</groupId>
    <artifactId>[my plugin artifactId]</artifactId>
    <version>[my plugin version]</version>

    <packaging>maven-plugin</packaging>

    <properties>
        <oo-sdk.version>[THE LATEST HPE OO_SDK VERSION]</oo-sdk.version>
        <oo-dotnet.version>[THE LATEST HPE OO_DOTNET VERSION]</oo-dotnet.version>
    </properties>

    <dependencies>
         <!-- required dependencies -->
         <dependency>
            <groupId>com.hp.oo.sdk</groupId>
            <artifactId>oo-legacy-action-plugin</artifactId>
            <version>${oo-sdk.version}</version>
         </dependency>
         <!-- end of required dependencies -->

         <dependency>
            <groupId>[groupId-1]</groupId>
            <artifactId>[artifactId-1]</artifactId>
            <version>[version-1]</version>
         </dependency>

         <dependency>
            <groupId>[groupId-2]</groupId>
```

```
            <artifactId>[artifactId-2]</artifactId>
            <version>[version-2]</version>
         </dependency>

         ...

        <dependency>
            <groupId>[groupId-n]</groupId>
            <artifactId>[artifactId-n]</artifactId>
            <version>[version-n]</version>
        </dependency>
    </dependencies>


    <build>
         <plugins>
            <plugin>
                <groupId>com.hp.oo.sdk</groupId>
                <artifactId>oo-action-plugin-maven-plugin</artifactId>
                <version>${oo-sdk.version}</version>
                <executions>
                   <execution>
                    <id>generate plugin</id>
                    <phase>process-sources</phase>
                       <goals>
                            <goal>generate-legacy-plugin</goal>
                       </goals>
                    </execution>
                </executions>
             </plugin>
         </plugins>
    </build>
</project>
```

Example: In the following example:

- The POM file is named **example.pom.xml**.
- The my-legacy-actions.jar contains the desired actions.
- The generated Maven plugin is `com.example:my-legacy-actions:1.0`.

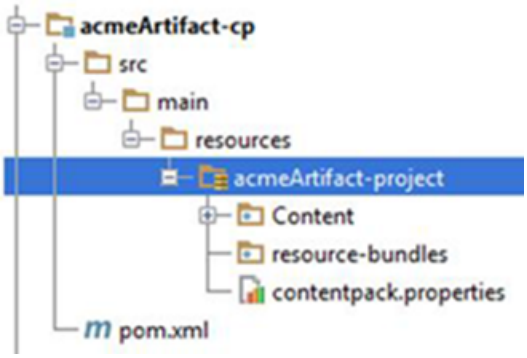
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                         http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>
```

```
    <groupId>com.example</groupId>
    <artifactId>my-legacy-actions-plugin</artifactId>
    <version>1.0</version>

    <packaging>maven-plugin</packaging>

    <properties>
         <oo-sdk.version>[THE LATEST HPE OO_SDK VERSION]</oo-sdk.version>
         <oo-dotnet.version>[THE LATEST HPE OO_DOTNET VERSION]</oo-dotnet.version>
    </properties>

    <dependencies>
         <!-- required dependencies -->
         <dependency>
            <groupId>com.hp.oo.sdk</groupId>
            <artifactId>oo-legacy-action-plugin</artifactId>
            <version>${oo-sdk.version}</version>
         </dependency>
         <!-- end of required dependencies -->

         <dependency>
            <groupId>com.example</groupId>
            <artifactId>my-legacy-actions</artifactId>
            <version>1.0</version>
         </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>com.hp.oo.sdk</groupId>
                <artifactId>oo-action-plugin-maven-plugin</artifactId>
                <version>${oo-sdk.version}</version>
                <executions>
                    <execution>
                        <id>generate plugin</id>
                        <phase>process-sources</phase>
                        <goals>
                            <goal>generate-legacy-plugin</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Creating Operations from @Actions

To create an operation from the @Action that you developed in the Developing Plugins for Java Actions section, you have to complete the following steps:

1. Find the Studio project inside the content pack module, for example, **acmeArtifact-cp/src/main/resources/acmeArtifact-project**.



2. Import this project into Studio.
3. Build the **acmeArtifact** parent module using Maven.
4. Import **acmeArtifact-plugin** to Studio.

   The location of the plugin that you have to import is: **%HOMEPATH%\.m2\repository\acmeGroup\acmeArtifact-plugin\1.0.0\acmeArtifact-plugin-1.0.0.jar**.

5. In Studio, create a new operation, and select your plugin in the **Create Operations** dialog box.

   Note: If you encounter errors when creating a new operation due to some missing dependencies, perform the following steps:

   a. Import the content pack from location: **%HOMEPATH%\.m2\repository\acmeGroup\acmeArtifact-cp\1.0.0\acmeArtifact-cp-1.0.0.jar**.

   b. Close the content pack in Studio.

   c. Retry to create the operation.

> These steps also apply for .NET and legacy Java actions.

Create a Content Pack

After creating an operation, you must create a content pack from your project, in order to deploy and use it in Central. You have three possible approaches to create the content pack:

- Create a content pack with Studio: see *Studio Guide* for details.
- Create a content pack with Maven:
  - Navigate to the location of the parent project, in the example in section Open the Project in a Java IDE, **c:\temp\acmeArtifact**, open the command line and `run mvn clean install`.
  - The content pack will be created inside the target folder from your content pack project. Open the Project in a Java IDE, **c:\temp\acmeArtifact\acmeArtifact-cp\target\acmeArtifact-cp-1.0.0.jar**)
- Create a content pack with OOSHA: see OOSHA Guide for details.

For details on how to deploy the content pack in Central, see *Central Guide.*

CloudSlang Content Development

Maven Artifacts

To create new CloudSlang content, you need to develop Java actions, grouped under a Maven Artifact.

In CloudSlang, actions are methods annotated with the @Action annotation.

> CloudSlang does not support .NET or legacy Java actions.

Developing Maven Artifacts for Java Actions

Preparing to Create a Maven Artifact

Install Maven

See the Preparing to Create a Plugin Using a Maven Archetype section from for a complete guideline on how to install and configure Maven.

> Install Maven 3.3.9 to develop actions for CloudSlang

Creating a Maven Project

1. Navigate to the location on the filesystem where your project should reside and start a shell in that directory.
2. On your command line, execute the following Maven command:

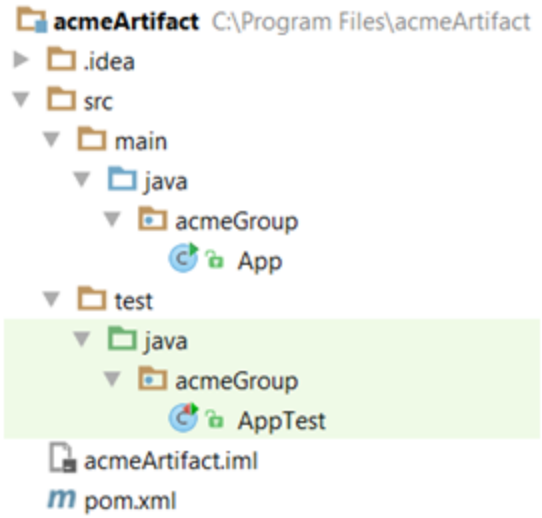
   ```
   mvn archetype:generate -DarchetypeArtifactId=maven-archetype-quickstart
   ```
3. During the project creation, enter the following details and press **Enter** after each one:
   - `groupId`: The group id for the resulting Maven project. `acmeGroup` is used in the example below.
   - `artifactId`: The artifact id for the resulting Maven project. `acmeArtifact` is used in the example below.
   - `version`: The version of the resulting Maven project. `1.0-SNAPSHOT` is used in the example below.
   - `package`: The package for the files in the project. The default for this option is the same as the `groupId`.
4. Locate the **pom.xml** file inside your project. Open it, and add the dependency below inside the `<dependencies>` tag.

   ```
   <dependency>
       <groupId>com.hp.score.sdk</groupId>
       <artifactId>score-content-sdk</artifactId>
       <version>1.10.6</version>
   </dependency>
   ```

Open the Project in a Java IDE

Open this project in a Java IDE application. It is recommended to use a Java IDE with Maven support.

The project contains the following directory structure:



Creating a New Action

A Java action is a method that conforms to the signature `public Map<String, String> doSomething(parameters)`, annotated with the **@Action** annotation.

You can find the complete specification of the **@Action** annotation in the Developing Actions section.

Build Maven Artifact

To build your maven project, navigate to the location of the project, for example, **c:\Program Files\acmeArtifact**, open the command line and run `mvn clean install`.

You can see the artifacts resulted from the build process under the target folder generated inside your project.

These artifacts are also installed in the local maven repository located at **%HOMEPATH%\.m2\repository or $HOME/.m2/repository**.

For example, you can find the resulted artifact from building `acmeArtifact` located in: **%HOMEPATH%\.m2\repository\acmeGroup\acmeArtifact\1.0-SNAPSHOT\acmeArtifact- 1.0-SNAPSHOT.jar**.

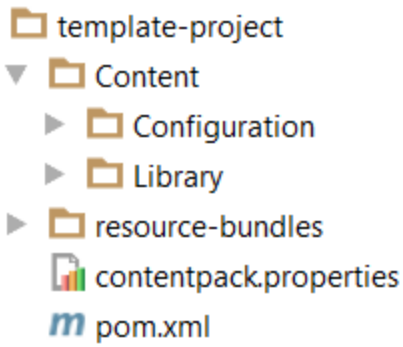
Creating CloudSlang Operation from Java Action

To create a CloudSlang operation, you must first create a project for placing the operation.

Creating a new Project from a predefined template

To create a new project:

1. Download **template-project.zip** from HPE LN.
2. Unzip **template-project.zip** to the desired location.
3. Find the directory **template-project** with the following structure.



4. Rename **template-project** directory to the name you want to provide for the new project

   For a **CloudSlang** project, the project naming convention is: `cs-[<vendor>]-<product|technology>`

   For example: `cs-vmware-vcenter`

5. Locate **pom.xml** file. Open it and replace the value inside the `artifactId` tag with the name of the project.

   For example: `<artifactId>cs-vmware-vcenter </artifactId>`

6. Locate the contentpack.properties file. Open it and provide values for the properties specified in that file.
   - **content.pack.name**: It is recommended to specify a value for this property. The value should be the name of the project.
   - **content.pack.version**: It is recommended to specify a value for this property. The value must be the same as the value from the version tag inside the **pom.xml** file. If you want to update the version of the project, you must update it inside both files.
   - **content.pack.description**
   - **content.pack**
   - **publisher**

Create CloudSlang Operation

Locate the Library folder inside your project, at `<project_path>/Content/Library`.

Inside the Library folder create a directory structure, representing the **namespace** of the operation.

> The **namespace** for a CloudSlang operation has a similar role to a Java package.

Example: Create the **namespace acme/operations**

Inside your **namespace**, create a new file with the **.sl** extension.

Example: The path of the new file will be **<project_path>/Content/Library/acme/operations/my_op.sl**.

Inside the **.sl** file, create a CloudSlang operation that calls the Java action. See the CloudSlang tutorial on how to build an operation for details.

Developing Maven Artifacts for Python Scripts

Create Maven Project

1. Create a Python project having the following tree structure:

```
my-python-module
|--__init__.py
|--another-module
|--__init__.py
|--script1.py
|--script2.py
```

2. Place the two template files, **pom.xml** and **assembly.xml** inside **my-python-module** folder.
3. Edit the **pom.xml** file and change the GAV of the module.

```
<groupId>mycompany</groupId>
<artifactId>my_first_artifact</artifactId>
<version>87.0</version>
```

Open Project in Python IDE

Open the project in a Python IDE, to create Python functions that will be called from Python operations.

Build Maven Artifact

To build your maven project, navigate to the location of the Python project (my-python-module path), open the command line and run `mvn clean install`.

You can see the artifacts resulted from the build process under the target folder generated inside your project.

These artifacts are also installed in the local maven repository located at **%HOMEPATH%\.m2\repository or $HOME/.m2/repository**.

For example, you can find the resulted artifact from building **my-python-module** located in: **%HOMEPATH%\.m2\repository\ mycompany \my_first_artifact\87.0\ my_first_artifact-87.0.zip**.

Creating CloudSlang Python Operation

To create a CloudSlang operation, you must first create a project for placing the operation.

You can create the project from a predefined template as described in section Creating CloudSlang Operation from Action.

Create CloudSlang Operation

Locate the Library folder inside your project, at **<project_path>/Content/Library**.

Inside the Library folder create a directory structure, representing the namespace of the operation.

> The namespace for a CloudSlang operation has a similar role to a Java package.

Example: Create the **namespace acme/operations**

Inside your **namespace**, create a new file with the **.sl** extension.

Example: The path of the new file will be **<project_path>/Content/Library/acme/operations/my_op.sl**.

Inside the **.sl** file, create a CloudSlang operation that calls a Python script. In the Python script, you can invoke the Python functions that you developed. See the CloudSlang tutorial on how to build a Python operation for details.

Create a Content Pack

After creating an operation, you must create a content pack from your project, in order to deploy and use it in Central. You can create a content pack from a CloudSlang project using OOSHA. See OOSHA Guide for details.

For details on how to deploy the content pack in Central, see *Central Guide.*

> Python libraries will not be included in the content pack built by OOSHA. In order to include them, you must copy the zip resulted from building your project inside the content pack Lib folder.

For example, you will have to copy the zip from **%HOMEPATH%\.m2\repository\ mycompany \my_ first_artifact\87.0\ my_first_artifact-87.0.zip** at the following location inside the content pack created by OOSHA: **Lib/mycompany/my_first_artifact/87.0/my_first_artifact-87.0.zip**.

Developing @Actions

An action is a method that has the `@com.hp.oo.sdk.content.annotations.Action` annotation.

@Action must be applied on Java methods that conforms to the signature:

```
@Action
public Map<String, String> doSomething(parameters)
```

The @Action annotation is invoked during flow execution, and specifies the following action information:

- **name**: name of the action
- **outputs**: action outputs
- **responses**: action responses

"Hello World!" Example

Following is a simple **"Hello World!"** action example:

```
public class MyActions {
     @Action
     public void sayHello() {
          System.out.println("Hello World!");
     }
 }
```

By default, the created @Action is named after the method that defines it. In the "Hello World!" example, the @Action name is **sayHello**. The @Action name is used in the operation's definition. The operation is the means to expose an @Action to Studio and to flow authors. Each operation points to a specific groupId, artifactId, version, and @Action name (GAV+@Action name).

You can customize the @Action name and provide a name that is different from the method name. You can do this using the @Action annotation value parameter. The following code defines the same "Hello World!" @Action, but names it my-hello-action:

```
public class MyActions {
    @Action(“my-hello-action”)
    public void sayHello() {
        System.out.println("Hello World!");
    }
 }
```

Passing Arguments to @Actions

An @Action is exposed to the flow context and can request parameters from it. The flow context holds the state of the flow. For example, consider the following @Action, which adds two numbers and prints the result to the console:

```
@Action
 public void sum(int x, int y){
   System.out.println(x+y);
 }
```

Parameters are taken from the context by name. The sum method requests two integer parameters x and y from the context. When invoking the @Action, HPE Operations Orchestration assigns the value of x and y from the context to the method arguments with the same name.

Just like with @Action, it is possible to customize parameter names and request that HPE Operations Orchestration resolves the value while using a custom name. In the following example, the sum method requests that the context op1 parameter is assigned to the x argument and op2 to the y argument:

```
@Action
 public void sum(@Param("op1") int x, @Param("op2") int y){
   System.out.println(x+y);
 }
```

The classes `ResponseNames`, `ReturnCodes`, `InputNames`, and `OutputNames`, under the `com.hp.oo.sdk.content.constants` package, include commonly used constants, which you can use in the @Action.

For example, input names such as HOST, USERNAME, PASSWORD, PORT, and so on, or response names such as SUCCESS, FAILURE, NO_MORE, and so on.

Return Values

An @Action, like any Java method, can also return a single value. The returned value is considered the return result of the @Action and is used as return result in the operation. It is also possible for an @Action to return multiple results to the operation. This is done by returning a Map<String, String>, where the Map key is the name of the result, and the associated value is the result value. Returning a Map<String, String> is a way for an @Action to pass multiple outputs to the operation at runtime.

Adding @Action Annotations

@Action annotations are used tso generate new operations in the Studio. When generating an @Action based operation, the new operation's initial attributes (description, inputs, outputs, responses) are taken from the @Action annotations definitions.

When developing plugins, you must correctly annotate the actions that return only a single value. The annotation has to declare an output with the special name `singleResultKey`. There is a constant `ActionExecutionGoal.SINGLE_RESULT_KEY` that assists you, for example:

```
@Action(name = "modulo-ten",
   description = "returns the last digit",
   outputs = @Output(ActionExecutionGoal.SINGLE_RESULT_KEY),
   responses = @Response(text = ResponseNames.SUCCESS,
   field = OutputNames.RETURN_RESULT,
   value = "0", matchType = MatchType.ALWAYS_MATCH,
   responseType = ResponseType.RESOLVED)
 )
 public int moduloTen(@Param("number") int number) {
   return number % 10;
 }
```

> It is important that you use @Action annotations; otherwise, operations created from these @Actions are harder to use.

Annotations

Adding metadata means adding or setting the relevant annotations and their attributes. The following table describes the @Action, @Output, @Response and @Param annotations:

Action

The `@com.hp.oo.sdk.content.annotations.Action` annotation specifies information on an action.

Attributes:

- value (optional): the name of the @Action
- description (optional)
- Output[] (optional): array of outputs (see below)
- Response[] (optional): array of responses (see below)

Comments:

You have two options for setting the name of the @Action:

1. The value attribute:

   ```
   @Action("aflPing") public void ping(…)
   ```

   or

   ```
   @Action(value="aflPing") public void ping(…)
   ```

2. The method name:

   ```
   @Action
    public void ping(…)
   ```

The names are checked in the above order. The first one checked is the value attribute. If it doesn't exist, the method name is selected.

Param

The `@com.hp.oo.sdk.content.annotations.Param` annotation specifies information on a parameter of an action.

Attributes:

- value: the name of the input
- required (optional): by default is false
- encrypted (optional): by default is false
- description (optional)

Comments:

This is important not only for the @Action data, but also for execution.

Inputs give an operation or flow the data needed to act upon. Each input is mapped to a variable. You can create an input for a flow, operation, or step.

In Studio, inputs can be:

- Set to a specific value
- Obtained from information gathered by another step
- Entered by the person running the flow, at the start of the flow

Output

The `@com.hp.oo.sdk.content.annotations.Output` specifies an output for an action.

Attributes:

- value: the name of the output
- description (optional)

Comments:

In order for the operation in Studio to have multiple outputs, the @Action itself has to declare them. Assigning values to multiple outputs can be achieved by creating an @Action whose return value is a Map<String, String>.

In order for the operation in studio to have only one output, the @Action itself has to declare it in the return value, and use the `SINGLE_RESULT_KEY` for binding.

The output is the data produced by an operation or flow. For example, success code, output string, error string, or failure message.

In Studio, the different kinds of operation outputs include:

- Raw result: the entire returned data (return code, data output, and error string).
- The primary and other outputs, which are portions of the raw result.

Response

The `@com.hp.oo.sdk.content.annotations.Action` annotation specifies a possible response of an action.

Attributes:

- `text`: the text displayed by each response transition
- `field`: the field to evaluate
- `value`: the expected value in the field
- `description`: (optional)
- `isDefault`: Indicates whether this is the default response. The default value is false. Only one response in a @Action can have this set to true.
- `mathType` : The type of matcher to activate against the value. For example if we defined (field = fieldName, value = 0, matchType = COMPARE_GREATER) this means that this response will be chosen if the field fieldName will have a value greater than 0.
- `responseType`: The type of the response (Success, Failure, Diagnosed, No_Action or Resolve_ By_Name).
- `isOnFail`: Indicates whether this is the On-Fail response. The default value is false. Only one response in a @Action can have this set to true.
- `ruleDefined`: Indicates whether or not this response has a rule defined. Responses that have no rules defined can be used as the default response. There should be only one response without a rule defined in a single @Action.

Comments:

A response is the possible outcome of an operation or flow. The response contains a single rule: field matches value.

@Action Data Definition Example

```
@Action(value = "aflPing",
        description = "perform a dummy ping",
        outputs = {@Output(value = RETURN_RESULT, description ="returnResult
```

Develop

```
description"),
                    @Output(RETURN_CODE),
                    @Output("packetsSent"),
                    @Output("packetsReceived"),
                    @Output("percentagePacketsLost"),
                    @Output("transmissionTimeMin"),
                    @Output("transmissionTimeMax"),
                    @Output("transmissionTimeAvg")},
        responses = {@Response(text = "success", field = RETURN_CODE, value =
PASSED),
                     @Response(text = "failure", field = RETURN_CODE, value =
FAILED)})

public Map<String, String> doPing(
   @Param(value = "targetHost",
         required = true,
         encrypted = false,
         description = "the host to ping") String targetHost,
   @Param("packetCount") String packetCount,
   @Param("packetSize") String packetSize) { …
}
```

Testing Extensions

As an @Action is a simple Java method, it is possible to test it using standard Java test tools such as JUnit, leveraging the normal lifecycle phases of a Maven project.

As the @Action itself is a regular method, it does not require invoking any HPE Operations Orchestration components. The invocation can be a direct Java method invocation in the test case.

Testing Extensions as Part of the Project Build

Once they are packaged into a plugin, you can invoke extensions from the command line for test purposes. The following is an @Action example:

```
public class TestActions {
    @Action
    public int sum(@Param("op1") int x, @Param("op2") int y){
        return x+y;
    }
}
```

Suppose the TestActions class is in a plugin with the following groupId, artifactId and version (GAV): com.mycompany:my-actions:1.0

You can invoke the sum @Action from the command line as follows:

```
mvn com.mycompany:my-actions:1.0:execute -Daction=sum -Dop1=1 -Dop2=3 -X
```

The result of this command is a long trace. The -X option is required to see log messages. Towards the end of the trace you can see:

[DEBUG] Configuring mojo 'com.mycompany:my-actions:1.0::execute' with basic configurator --> [DEBUG] (f) actionName = sum [DEBUG] (f) session = org.apache.maven.execution.MavenSession@21cfa61c [DEBUG] -- end configuration -- [DEBUG] Action result: action result = 4

References

- Maven Getting Started Guide: https://maven.apache.org/guides/getting-started/index.html
- Installing Apache Maven: https://maven.apache.org/install.html
- Using Mirrors for Repositories: https://maven.apache.org/guides/mini/guide-mirror-settings.html
- CloudSlang Tutorial: http://cloudslang-docs.readthedocs.io/en/v1.0/section_tutorial.html

Hewlett Packard
Enterprise