HP Operations Orchestration

For the Windows and Linux

Software Version: 10.02

Application Program Interface (API) Guide

Document Release Date: January 2014

Software Release Date: January 2014

Legal Notices

Warranty

The only warranties for HP 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. HP 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 HP 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.

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

AMD and the AMD Arrow symbol are trademarks of Advanced Micro Devices, Inc.

Google™ and Google Maps™ are trademarks of Google Inc.

Intel®, Itanium®, Pentium®, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and other countries.

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

Microsoft®, Windows®, Windows NT®, Windows® XP, and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.

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

UNIX® is a registered trademark of The Open Group.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: **http://h20230.www2.hp.com/selfsolve/manuals**

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to: **http://h20229.www2.hp.com/passport-registration.html**

Or click the **New users - please register** link 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 HP sales representative for details.

Support

Visit the HP Software Support Online web site at: **http://www.hp.com/go/hpsoftwaresupport**

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

HP Software Solutions Now accesses the HPSW Solution and Integration Portal Web site. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this Web site is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

Contents

Contents 3
Introduction 7
Basic Concepts 7
RESTful APIs 7
Request Headers 7
Integration Use Case 8
Use Case Description 8
Use Case Implementation 8
Portal/App Admin Process - Selecting Flows to Expose to Users 8
End User process - Invoking and Monitoring Workflows 11
Basic Authentication 13
Backward Compatibility With HP OO 9.x APIs 14
SOAP 14
REST 19
REST APIs 20
Flow Execution 21
Get Execution 21
Get Execution Summary 25
Retrieve Feed Events 26
Execution Pauses 32
Get Execution Log 36
Change the Status of an Execution 38
Execute a Flow by UUID 41
Ad-hoc Flow Execution 42
Flow Execution Status 43
Flow Input 44
Flow Library 47
Read Next Level of Library Tree 47
Get Partial Tree 48

Find Tree Item By Path 49
Get Flow Details 50
Get Flow Inputs 51
Scheduler 53
Create New Flow-Schedule 53
Enable Flow-Schedule 55
Delete Flow-Schedule 56
Get Flow-Schedules 56
Get Flow-Schedule Details 57
Update Flow-Schedule 58
Dashboard 59
Get Statistics 59
Deployment 61
Deploy Content Packs 61
How can I deploy content packs with progress? 63
Retrieve Details for Deployed Content Packs 63
Roll Back Content Packs 64
Create Deployments 65
Upload the Deployment Process File 66
Run a Specific Deployment Process 67
Get the Deployment Process Object 68
Delete Content Pack from Deployment Process 70
Content Configurations 71
Create Content Configuration 71
Delete Content Configuration 72
Get All Content Configurations 72
Get Content Configuration 73
Update Content Configuration 74
Group Aliases 75
Create a Workers Group Alias 75
Get All Groups Aliases 76

Get Group Alias by Name ... 76
Delete Group Aliases ... 77
Update a Group Alias ... 78
Workers ... 79
Update a Specific Worker ... 79
Get All Workers ... 79
Get All Workers Groups ... 80
Assign Workers to a Workers Group ... 81
Remove Workers from a Workers Group ... 81
LDAP Configuration ... 82
Create a New LDAP Configuration ... 82
Update an Existing LDAP Configuration ... 84
Delete an LDAP Configuration ... 86
Get LDAP Configurations ... 87
Test User Attributes ... 88
Test User Groups ... 90
Test User Attributes with Existing LDAP Password ... 91
Test User Groups with Existing LDAP Password ... 93
Users ... 94
Create New Internal User ... 94
Update Existing Role ... 96
Delete an Internal User ... 97
Get Users ... 97
Get Session's User ... 98
LW SSO ... 99
Get LW SSO Configuration ... 99
Update LW SSO configuration ... 100
Authentication ... 101
Get Authentication Configurations ... 101
Update Authentication configurations ... 101
Roles ... 102

Get Specified Role 102
Get All Roles 103
Create New Role 105
Update an Existing Role 108
Delete a Role 109
Get the Default Role 109
Update the Default Role 110
Get Entitlements Per Path and Roles 110
Update Path Entitlement Per Role 111
Configuration Items 113
Create a Configuration Item 113
Get All Configuration Items 113
Get a Configuration Item 114
Update Configuration Item 114
Miscellaneous 116
Get HP OO Version 116

Introduction

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

The public API is HTTP-based.

All APIs are RESTful and use JavaScript Object Notation (JSON).

Note: HP OO version 10.10 or later 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!”
}
```

Basic Concepts

See the Concepts Guide for more information on the basic concepts of HP Operations Orchestration.

RESTful APIs

All REST APIs have a prefix of `/rest`. For example, `POST/rest/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.

Some APIs provide `application/rss+xml` or `application/atom+xml`.

Integration Use Case

This chapter describes a common usage of the HP 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 HP OO platform integration. HP OO APIs allow much more than that.

Use Case Description

The most common use case when integrating with HP 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

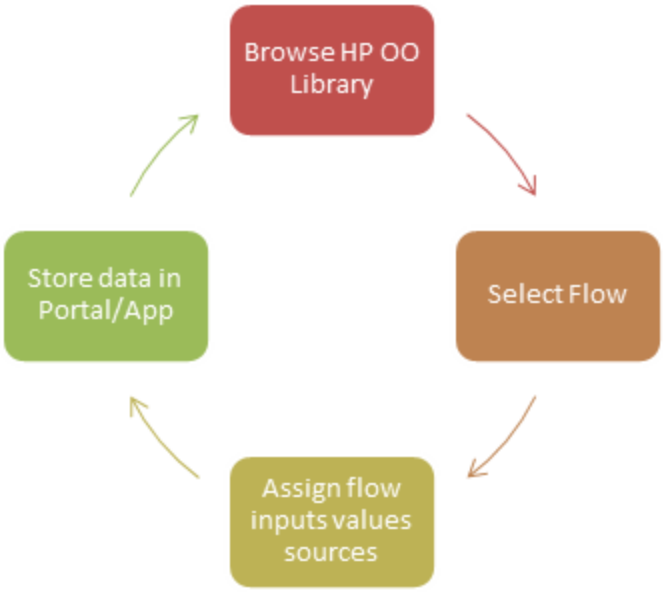
The integration includes two separate processes. These processes are described from the user perspective, but also describe the work to be done by the integration developer.

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 the 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 HP OO Central server look like.

Step: Browse HP 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 HP 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

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

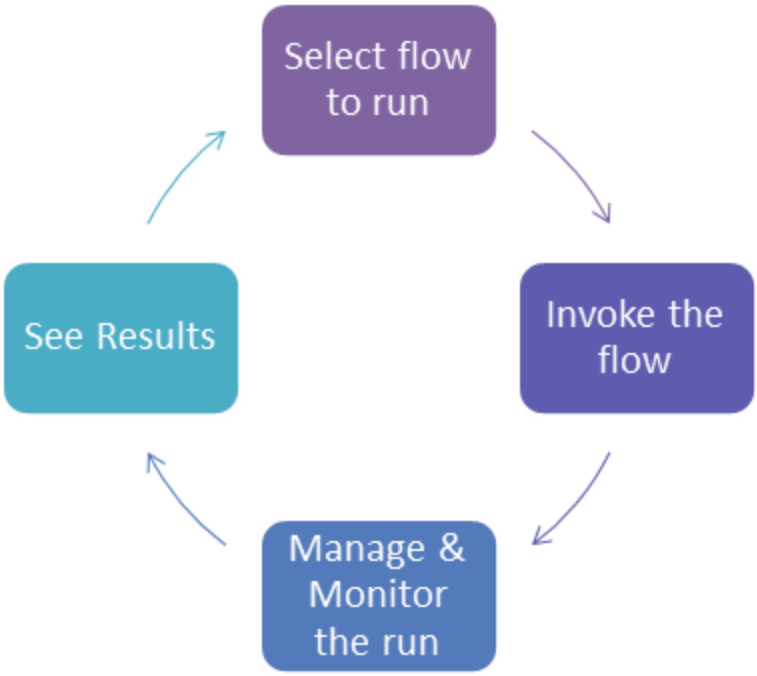
Admin Action: Store all the information in the Portal/Application.

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 HP OO Central server looks like.

The interaction is done through the HP OO REST API.

Refer to "Flow Execution" on page 21 for more technical 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 `logLevel` and `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`

Step: Manage and Monitor the run.
The main difference between these two options is the data included. Option 1 includes only general data of the run status while option 2 contains the raw data of the execution in RSS format. In this option the integrator should use the following tags under each entry (event):

- `<title>`: Contains the title of the event

  For Example: `<title>Execution started</title>`. The user can see the status of the run in the **portal/application**.

- `<content>`: A JSON format step execution data that can be consumed and formatted to display to the user.

  For example:

  ```
  <content type="text">{"execution_name":"How do I- Create a parallel flow","tri
  gger_type":"MANUAL","flow_UUID":"d012e1c3-704f-426f-a380-b2425a166d39","EXECUT
  ION_EVENTS_LOG_LEVEL":"DEBUG"}</content>
  ```

  See the Execution Event Log for more information.

Integrator Actions (interaction with OO): There two options to do that:

1. The code will have a loop that continuously calls HP OO to get the status.
2. The code will need to implement RSS reader (using existing libraries available on the market) and use it to listen to the event feed from HP OO (will also include a loop).

The main difference between the two options is the included data. Option **2** contains the raw data of the execution under the `<content>` tag, which is in JSON format and can be consumed and formatted to display to the user, while option **1** includes only general data of the run status.

API to use:

1. Get Flow Execution Summary `GET/executions/{executionId}/summary`
2. Retrieve Feed Events `GET/executions/{executionId}`

- **End User Action**: Control the run (Optional).

Users can take the following actions on the run:

1. Pause the run.
2. Resume the run.
3. Cancel the run.

Integrator Actions (interaction with OO): Implementing some or all of this will provide more control to the end user, which can be very helpful to some end user types. But on the other hand need to have additional UI development on the portal/application side. When implementing `Pause and Resume` make sure to keep track on the run status after Resume action was activated.

API to use:

- See Pause: `GET/executions/{executionId}/pause`
- See Resume: `PUT/executions/{executionId}`
- Cancel

Step: Get Results.

End User Action: Expose to the user the final result for the flow and maybe even include the raw data returned from it.

Integrator Actions (interaction with OO): Here the code made before for tracking the status using RSS feed can be reused. The result of the flow get be retrieved from the last event:
`<title>Flow execution finished</title>`
And use the `<content>` tag.

API to use: Retrieve Feed Events (RSS): `GET/executions/{executionId}`

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.

Backward Compatibility With HP OO 9.x APIs

Some SOAP and REST APIs from HP OO 9.x are supported by HP OO 10.x, and some are not.

Some of the APIs from HP OO 9.x have equivalent REST APIs for HP OO 10.x. We recommend using the REST APIs for HP OO 10.x.

- The base path for using HP OO 10.x REST API is: http(s)://<OO Central Server Name / IP>:<PORT>/oo/rest/.
- The URL for using HP OO 9.x SOAP API while working with OO 10.x Central is the same as in HP OO 9.x. That is, https:// <OO Central Server Name / IP>:<PORT>/PAS/services/WSCentralService.
- The URL for using HP OO 9.x REST API while working with OO 10.x Central is the same as in HP OO 9.x. That is, https:// <OO Central Server Name / IP>:<PORT>/PAS/services/rest.

SOAP

Below you can find information on what is supported, what is not, and the HP OO 10.x API that we recommend to use. For details on the HP OO 10.x REST requests, see the section below.

Functionality	9.x Request	10.x Support for 9.x Request	10.x Equivalent REST Request
Configurations	getLWSSOConfig	Not Supported	GET/authns/lwsso-config
	updateLWSSOConfig	Not Supported	PUT/authns/lwsso-config
Clusters	getClusterNodes	Not Supported	N/A

Functionality	9.x Request	10.x Support for 9.x Request	10.x Equivalent REST Request
Flows	getFlowDetails	Supported	GET/flows/{uuid}
	getFlowGraph	Partially Supported. The request will succeed, but a static image is returned saying that this feature is not supported.	N/A
	getFlowInputDescriptions	Not Supported	GET/flows/{uuid} /inputs
Groups and User Management **Note:** In HP OO 10.x, user groups are called user roles.	createGroup	Not Supported	POST/roles
	updateGroup	Not Supported	PUT/roles/ {roleName}
	deleteGroup	Not Supported	DELETE/roles/ {roleName}
	getUserGroups	Not Supported	GET/roles
	createUser	Not Supported	POST/users
	updateUser	Not Supported	PUT/users/ {username}
	deleteUser	Not Supported	DELETE/users/ {userIds}

Functionality	9.x Request	10.x Support for 9.x Request	10.x Equivalent REST Request
Repositories **Note:** In HP OO 10.x, the concept of repository was replaced with new concepts. See the *HP OO Concepts Guide*.	getPermissions	Not Supported	In order to control content permissions, use: GET/roles/{rolesNames}/entitlements/** or PUT/roles/{roleName}/entitlements/**.
	setPermissions	Not Supported	
	getAttributes	Not Supported	
	renameRepoEntity	Not Supported	
	deleteRepoEntity	Not Supported	
	moveFlow	Not Supported	
	updateDescription	Not Supported	
	createFolder	Not Supported	
	moveFolder	Not Supported	
	list	Supported	GET/flows/tree GET/flows/tree/sub GET/flows/tree/level
	search	Supported	N/A

Functionality	9.x Request	10.x Support for 9.x Request	10.x Equivalent REST Request
Runs	getFlowsRunHistory	Not Supported	N/A
	getFlowRunHistory	Supported	GET/executions
	pauserun	Supported	PUT/executions/{executionId}/status
	resumerun	Supported	PUT/executions/{executionId}/status PUT/executions/{executionId}/status
	cancelrun	Supported	PUT/executions/{executionId}/status
	runFlow	Supported	POST/executions
	runFlowEx	Supported	POST/executions
	getRunStatus	Supported	GET/executions/{id} GET/executions/{executionIds}/summary
	getRunStatusEx	Supported	GET/executions/{id} GET/executions/{executionIds}/summary
	getStatusForRuns	Not Supported	N/A

Functionality	9.x Request	10.x Support for 9.x Request	10.x Equivalent REST Request
Scheduler	isScheduledFlowPaused	Not Supported	GET/schedules/ GET/schedules/{id}
	isSchedulerPaused	Not Supported	GET/schedules/ GET/schedules/{id}
	isSchedulerEnabled	Not Supported	GET/schedules/ GET/schedules/{id}
	getSchedulesForFlowCategory	Not Supported	N/A
	pauseScheduledFlow	Not Supported	/PUT/schedules/{ids}/enabled
	pauseSchedule	Not Supported	PUT/schedules/{ids}/enabled
	resumeSchedule	Not Supported	PUT/schedules/{ids}/enabled
	scheduleFlow	Not Supported	POST/schedules
	getSchedule	Not Supported	GET/schedules/{id}
	deleteSchedule	Not Supported	DELETE/schedules/{ids}
	getScheduledFlows	Not Supported	GET/schedules
	getSchedulesOfFlow	Not Supported	GET/schedules
	resumeScheduledFlow	Not Supported	PUT/schedules/{ids}/enabled
	deleteScheduledFlow	Not Supported	GET/schedules DELETE/schedules/{ids}
Selection Lists	getSelectionList	Not Supported	N/A
	createSelectionList	Not Supported	N/A
Repositories	/list/{path}	Supported	N/A
Runs	/run/{flow path/uuid}	Supported	POST/executions

REST

Functionality	9.x Request	10.x Support for 9.x Request	10.x Equivalent REST Request
Repositories	/list/{path}	Supported	GET/flows/tree GET/flows/tree/sub GET/flows/tree/level
Runs	/run/{flow path/uuid}	Supported	POST/executions

REST APIs

This section includes the RESTful APIs used in HP Operations Orchestration version 10.02.

Flow Execution

These APIs enable you to execute flows.

Get Execution

Request: `GET/executions`

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 parameters:

Attribute	Type	Description	Required	Default Value
`pageNum`	Integer	Number of page to display.	Yes	
`pageSize`	Integer	The number of rows in the page.	Yes	
`flowPath`	String	The full path of the flow that was executed.	No	All flows
`status`	Array of Predefined Values	An array of execution statuses used for filtering. Possible values: `RUNNING, COMPLETED, SYSTEM_FAILURE, PAUSED, PENDING_PAUSE, CANCELED, PENDING_CANCEL` See "Flow Execution Status" on page 43.	No	All execution statuses
`owner`	String	Pattern of user that is the owner of the execution	No	All owners
`resultStatusType`	Array of Predefined Values	An array of result status types used for filtering. Possible values: `RESOLVED, DIAGNOSED, ERROR, NO_ACTION_TAKEN`	No	All result status types

Attribute	Type	Description	Required	Default Value
pauseReason	Array of Predefined Values	An array of pause reasons for filtering. Available only when PAUSED is in the list of execution statuses. Possible values: USER_PAUSE, INPUT_REQUIRED, SELECT_TRANSITION, DISPLAY, GATED_TRANSITION, HAND_OFF, INTERRUPT, NO_WORKERS_IN_GROUP, BRANCH_PAUSED	No	All pause reasons
startedBefore	Long	Timestamp in milliseconds used for filtering executions that started earlier than this time. Must be later than startedAfter, if both are used.	No	All starting times
startedAfter	Long	Timestamp in milliseconds used for filtering executions that started later than this time. Must be earlier than startedBefore, if both are used.	No	All starting times

Examples:

```
GET /executions?pageNum=1&pageSize=10&flowPath=025931b0-c7ff-445f-9690-43f136136c9a&status=COMPLETED&owner=admin&resultStatusType=RESOLVED&resultStatusType=DIAGNOSED&startedBefore=1385381554609&startedAfter=1385381449092
```

```
GET /executions?pageNum=3&pageSize=8&resultStatusType=PAUSED&pauseReason=USER_PAUSED&pauseReason=HAND_OFF
```

Response entity body:

An array which contains all of the executions which match the requested filters. Each element in the array represents an execution.

The returned array is ordered by descending starting times. Therefore, the execution that has the latest starting time will be first, and would be empty if no execution exists for the given filters.

Attribute	Type	Description	Comments
executionId	String	The ID of the execution.	

Attribute	Type	Description	Comments
branchId	String	The ID of the branch were a pause has occurred.	null when status is not PAUSED
startTime	Long	Timestamp in milliseconds when the execution started.	
endTime	Long	Timestamp in milliseconds when the execution ended.	null if execution hasn't ended
status	Predefined Value	The status of the execution. Possible values: RUNNING, COMPLETED, SYSTEM_FAILURE, PAUSED, PENDING_PAUSE, CANCELED, PENDING_CANCEL See "Flow Execution Status" on page 43.	
resultStatusType	Predefined Value	The result status type of the execution. RESOLVED, DIAGNOSED, ERROR, NO_ACTION_TAKEN	
resultStatusName	String	The name of the result status.	
pauseReason	Predefined Value	The reason for the pause. Possible values: USER_PAUSED, INPUT_REQUIRED, SELECT_TRANSITION, DISPLAY, GATED_TRANSITION, HAND_OFF, INTERRUPT, NO_WORKERS_IN_GROUP, BRANCH_PAUSED	Available only when status is PAUSED
owner	String	The user that most recently owned the execution (either triggered it, or resumed it).	
triggeredBy	String	The user that triggered the execution.	
flowUuid	String	The uuid of the flow which was executed.	
flowPath	String	The path of the flow which was executed.	
executionName	String	The name of the execution.	
roi	Double	The value of the ROI of the execution.	
branchesCount	Long	Deprecated.	Always returns 0

Example:


```
[
   {
      "executionId":"100749",
      "branchId":null,
      "startTime":1371106274153,
      "endTime":1371106277160,
      "status":"COMPLETED",
      "resultStatusType":"RESOLVED",
      "resultStatusName":"success",
      "pauseReason":null,
      "owner":"anonymousUser",
      "triggeredBy":"anonymousUser",
      "flowUuid":"06fe8531-868b-4e79-aa7a-13a5e30a66ec",
      "flowPath":"Library/Utility Operations/Samples/Generate/Number.xml",
      "executionName":"Generate Random Number",
      "branchesCount":0,
      "roi":null
   },
   {
      "executionId":"100267",
      "branchId":null,
      "startTime":1371104522563,
      "endTime":1371104576253,
      "status":"COMPLETED",
      "resultStatusType":"ERROR",
      "resultStatusName":"failure",
      "pauseReason":null,
      "owner":"anonymousUser",
      "triggeredBy":"anonymousUser",
      "flowUuid":"1901edde-3cac-4da6-915c-fd254e23169c",
      "flowPath":"Library/Multihost Connectivity Diagnostic.xml",
      "executionName":"Multihost Connectivity Diagnostic",
      "branchesCount":0,
      "roi":null
   }
]
```


Response status codes:

Code	Meaning	Returned When
200	OK	All requested executions were returned.
400	Bad Request	Invalid values assigned to `pageNum`, `pageSize`, `status`, `pauseReason`, `startedBefore` or `startedAfter` (if `startedAfter` is later than `startedBefore`).

Get Execution Summary

Request: `GET/executions/100976,100876/summary`

Description: Retrieves the details of a specific execution.

Example:

`/executions/3332190961082830376,679861347442169334/summary`

Request path variables:

Attribute	Description	Required
`executionIds`	The ids of the executions	Yes

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",
       "triggeredBy":"anonymous",
       "flowUuid":"a8e8fc10-b584-4d39-921f-987b29c9dd19",
       "flowPath":null,
       "executionName":"mock flow",
       "branchesCount":0,
       "roi":null
},
{
       "executionId":"679861347442169334",
       "branchId":null,
       "startTime":1371475041169,
```

```
        "endTime":null,
        "status":"PAUSED",
        "resultStatusType":"RESOLVED",
        "resultStatusName":"HAHA",
        "pauseReason":"USER_PAUSED",
        "owner":"anonymous",
        "triggeredBy":"anonymous",
        "flowUuid":"a8e8fc10-b584-4d39-921f-987b29c9dd19",
        "flowPath":null,
        "executionName":"mock flow",
        "branchesCount":0,
        "roi":null
}
]
```


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

Retrieve Feed Events

Request: `GET/executions/{id}`

Description: Get the flow execution events feed for the given execution ID (the result of the flow execution request).

Request path variables:

Attribute	Description	Required
`id`	The execution id of the executed flow	Yes

Request header:

The `content-type` is: `application/json`

The `accept` should be set according to the desired web feed format: `application/rss+xml` or `application/atom+xml`

Response status codes:

Code	Meaning	Returned When
200	OK	
404	Not Found	The requested execution id can't be found

Response entity body:

- **on success:**

  Returns a syndication feed in the required format, RSS or ATOM format. The return feed contains the execution events ordered by the flow execution sequence.

Each entry (ATOM format) in the feed or item (RSS format) is an event.

RSS supports version 2.0.

ATOM supports version 1.0

ATOM feed example:

```
<feed xmlns="http://www.w3.org/2005/Atom"xmlns:dc="http://purl.org/dc/elements/1.1/">
     <title>Flow Execution [0dbc2384-c97f-4eee-8e1d-1b4f43fdb47e]</title>
     <link rel="self" href="http://localhost:8080/oo/rest/executions/0dbc2384-c97f-4eee-8e1d-1b4f43fdb47e" />
     <subtitle>Flow execution events feed</subtitle>
     <id>urn:uuid:0dbc2384-c97f-4eee-8e1d-1b4f43fdb47e</id>
     <updated>2012-08-15T12:53:10Z</updated>
     <dc:date>2012-08-15T12:53:10Z</dc:date>
```

Following are the different events that the feed contains, sorted by the flow execution process:

State in Flow	Event Type	Title	Description	Content	Comment
Flow triggered	START	Execution started	Flow [UUID] execution running started	{ "flow_UUID": [flow_ UUID], "trigger_type": [trigger_type], "execution_name": [execution_name] }	trigger type: manual or scheduled
Flow triggered	FLOW_ INPUT	Flow input	[param_name] =[param_ value]	{ "param_name": [param_name], "param_value": [param_value] }	One for each flow input
Start flow execution	DEBUG LOG	Initialize Flow variables	Initialize Flow variables	flow_variables : [{ flow_ variable: flow_ value }*]*	
Enter step	INFO LOG	Start Step	Step ID and name	{ "step_id": [step_id], "step_ name": [step_ name] }	Step name may not exist
Before operation execution	INFO LOG	Step inputs	Step inputs after evaluation	step_inputs : [{ step_input : step_value }*]*	

State in Flow	Event Type	Title	Description	Content	Comment
Before operation execution	INFO LOG	Operation group	Operation group name	`{ "operation_group": [operation_group] }`	
during step execution	ERROR LOG	Execute step: operation error	Error occurred during operation execution	`{ "error_message": [error_message] }`	Exception during step execution
during step execution	ERROR LOG	Step execution: navigation error	Error occurred during navigation execution	`{ "error_message": [error_message] }`	Exception during step navigation
Before step ended	DEBUG LOG	Execute step: operation outputs	Operation additional outputs after Operation execution	`opertion_outputs : [ { operation_output : operation_value } * ]*`	During binding of the step result
Before step ended	DEBUG LOG	Execute step: raw outputs	Operation action raw results after Operation execution	`opertion_results: [ { operation_result: operation_value } * ]*`	
Before step ended	DEBUG LOG	Execute step: primary output	Operation primary output	`{ "primary_output": [primary_output] }`	
Before step ended	DEBUG LOG	Execute step: response	Operation response	`{ "response_name": [response_name ], "response_type": [response_type] }`	

State in Flow	Event Type	Title	Description	Content	Comment
Before step ended	INFO LOG	Execute step: results	Step results after step execution	`step_results : [ { step_result : step_value }* ]*`	Contains only the parameters that where added or updated during this step execution
Before step ended	DEBUG LOG	Execute step: transition	Step transition info	`{ "transition_ name": [transition_ name], "transition_ desc": [transition_ desc], "response_ name": [response_ name] }`	
Before step ended	INFO LOG	Execute step: primary result	Step primary result	`{ "primary_ result": [primary_result] }`	
Before sub flow started	DEBUG LOG	Start sub flow	Start sub flow		
Before sub flow ended	DEBUG LOG	End sub flow	End sub flow		
Before start branch	INFO LOG	Execute step: multi instance step start	Multi instance number of {instance_ num} instances started now	`{ "instance_num": [instance_num] }`	Split point - for multi steps
Before start branch	INFO LOG	Execute step: parallel step start	Parallel step number of { instance_ num} instances started now	`{ "instance_num": [instance_num] }`	Split point - for parallel

State in Flow	Event Type	Title	Description	Content	Comment
Before start branch	DEBUG LOG	Start Branch	Branch has started	`{ "branch_id": [branch_id] }`	
Before branch ended	DEBUG LOG	End Branch	Branch has ended	`{ "branch_id": [branch_id] }`	
Before branch ended	INFO LOG	Execute step: multi instance step end	Multi instance step merged all the instances		For multi steps - merge point
Before branch ended	INFO LOG	Execute step: parallel step end	Parallel step merged all the branches		For parallel steps – merge point
End flow execution	INFO LOG	Flow execution: outputs	Flow outputs	`flow_outputs : [ { flow_output : flow_value }* ]*`	
End flow execution	FLOW_ RESULTS	Flow execution: results	Flow execution running finished with result type [result type] and result name [result name]	`{ "result_name": [result _name ], "result_type": [result_type] }`	Result type is one of: resolved, error, no action taken or diagnosed
End flow execution	FINISH_ SUCCESS	Flow execution finished	Flow execution finished with status COMPLETED	`{ "execution_ status": [execution_ status] }`	Execution status is one of: completed, canceled or failure
End flow execution	FINISH_ FAILURE	Flow execution finished	Flow execution finished with status FAILURE	`{ "execution_ status: [execution_ status], "error_ message": [error_ message] }`	In case the execution ended with failure

State in Flow	Event Type	Title	Description	Content	Comment
End flow execution	FINISH_ CANCELED	Flow execution canceled	Flow execution finished with status CANCELLED	`{ "execution_ status": [execution_ status] }`	The execution was canceled by the user

Execution Pauses

Request:`GET/executions/{executionId}/pauses`

Description: Retrieves current pauses for the given execution id.

Request path variables:

Attribute	Type	Description	Required
`executionId`	String	The ID of the execution which the client wishes to retrieve its pauses.	Yes

Example:

```
GET/executions/100001/pauses
```

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 the FlowInput	

Example:

```
[
   {
      "pauseId":101100014,
      "executionId":"100100355",
      "branchId":null,
      "stepId":"fa351d72-d381-4159-b81b-5fff493f7e41",
      "stepName":"Parallel Split",
      "pauseReason":"INPUT_REQUIRED",
      "requiredInputs":[
         {
            "uuid":null,
            "name":"parallelInput",
            "valueDelimiter":null,
            "description":"Just checking",
            "encrypted":false,
            "multiValue":false,
            "mandatory":false,
            "sources":null,
            "type":"String",
            "validationId":null,
            "defaultValue":null
         }
      ]
   }
]
```


Reason 2: Display

Attribute	Type	Description	Comments
`pauseReason`	Predefined Value	The value `DISPLAY`	
`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 brach.
`stepId`	String	The UUID of the step in which the pause has occurred.	
`stepName`	String	The name of the step in the flow.	
`title`	String	The author's specified localized title for the prompt message that should be presented to the user.	

Attribute	Type	Description	Comments
text	String	The author's specified localized message that should be presented to the user	
height	String	For future use.	Ignore this field.
width	String	For future use.	Ignore this field.

Example:

```
[
   {
      "pauseId":101100010,
      "executionId":"100100312",
      "branchId":null,
      "stepId":"c12fc96a-f938-4a8c-81a1-5dbcc3b81e6d",
      "stepName":"Resolved : success",
      "pauseReason":"DISPLAY",
      "title":"Return Step title with input: value1",
      "text":"Return Step text:\ninput1: value1",
      "height":"0.0",
      "width":"0.0"
   }
]
```

Reason 3: Gated Transition

Attribute	Type	Description	Comments
pauseReason	Predefined Value	The value GATED_TRANSITION	
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 brach.
stepId	String	The UUID of the step in which the pause has occurred.	
stepName	String	The name of the step in the flow.	
roleName	String	The role which the user is missing in order to pass the gated transition.	
userName	String	The user which is trying to pass through the gated transition.	

Example:

```
[
   {
      "pauseId":101100008,
      "executionId":"100100274",
      "branchId":null,
      "stepId":"2ba782f2-ea84-4f66-936d-4226d4bfa134",
      "stepName":"UUID Generator",
      "pauseReason":"GATED_TRANSITION",
      "roleName":"AUDITOR",
      "userName":"anonymousUser"
   }
]
```


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 brach.
`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`	

Attribute	Type	Description	Comments
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 brach.
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

Example:

```
[
   {
      "pauseId":101100012,
      "executionId":"100100341",
      "branchId":null,
      "stepId":null,
      "stepName":null,
      "pauseReason":"NO_WORKERS_IN_GROUP",
      "groupName":"RAS_Operator_Path"
   }
]
```

Response status codes:

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

Get Execution Log

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

Description: Retrieves the details of a specific execution.

Request path variables:

Attribute	Description	Required
executionId	The id of the execution	Yes

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:


```
{
   "executionSummary":{
      "executionId":"348246628680024354",
      "branchId":null,
      "startTime":1371366300297,
      "endTime":null,
      "status":"PAUSED",
      "resultStatusType":"RESOLVED",
      "resultStatusName":"HAHA",
      "pauseReason":"USER_PAUSED",
      "owner":"anonymous",
      "triggeredBy":"anonymous",
      "flowUuid":"a8e8fc10-b584-4d39-921f-987b29c9dd19",
      "flowPath":null,
      "executionName":"mock flow",
      "branchesCount":0,
      "roi":null
   },
   "executionLogLevel":"INFO",
   "flowInputs":{
      "flowInput4":"flowInput4Value",
      "flowInput2":"flowInput2Value",
      "flowInput3":"flowInput3Value",
      "flowInput0":"flowInput0Value",
      "flowInput1":"flowInput1Value"
   },
   "flowVars":[
      {
         "name":"flowVar0",
         "termName":"flowVar0TermName",
         "value":"flowVar0Value"
      },
      {
         "name":"flowVar1",
         "termName":"flowVar1TermName",
         "value":"flowVar1Value"
      },
      {
```

```
            "name":"flowVar2",
            "termName":"flowVar2TermName",
            "value":"flowVar2Value"
         },
         {
            "name":"flowVar3",
            "termName":"flowVar3TermName",
            "value":"flowVar3Value"
         },
         {
            "name":"flowVar4",
            "termName":"flowVar4TermName",
            "value":"flowVar4Value"
         }
      ],
      "flowOutput":{
         "flowOutput4":"flowOutput4Value",
         "flowOutput3":"flowOutput3Value",
         "flowOutput0":"flowOutput0Value",
         "flowOutput2":"flowOutput2Value",
         "flowOutput1":"flowOutput1Value"
      }
   }
```


For more information, see the returned items in the Get Execution API.

Change the Status of an Execution

Request: `PUT/executions/{executionId}/status`

Description: Update an existing execution status.

Request path variables:

Attribute	Type	Description	Required
`executionId`	String	The ID of the execution which the user wants to update the status.	Yes

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.

To cancel an execution:

Attribute	Type	Description	Required	Default value
`action`	Predefined Value	The value `CANCEL`	Yes	

Example:

```
{
   "action":"CANCEL"
}
```

To pause an execution:

Attribute	Type	Description	Required	Default value
`action`	Predefined Value	The value `PAUSE`	Yes	

Example:

```
{
   "action":"PAUSE"
}
```

For reassigning an execution to another user

Attribute	Type	Description	Required	Default value
`action`	Predefined Value	The value `REASSIGN`	Yes	
`data`	Key value	Contains the key `userName` and the reassigned user as value.	Yes	

Example:

```
{
   "action":"REASSIGN",
   "data":{
      "userName":"John"
   }
}
```

For resuming an execution

Attribute	Type	Description	Required	Default value
`action`	Predefined Value	The value "REASSIGN"	Yes	

Attribute	Type	Description	Required	Default value
`data`	Key value	Should contain two key-value pairs: • `branchId` key for the branch that should be resumed. Null should be provided for the brach root. • `input_binding` key is used to provide the required inputs for an execution which is waiting for inputs. All the inputs should be provided within a String. (The desired JSON within ")		• `brachId` is required. • `input_binding` only required when resuming an execution which has the `pauseReason` `INPUT_REQUIRED`. See Execution Pauses

Examples:

- Resume without inputs:

```
{
   "action":"RESUME",
   "data":{
      "branchId":null
   }
}
```

- Resume with inputs:

```
{
   "action":"RESUME",
   "data":{
      "branchId":"f47ac10b-58cc-4372-a567-0e02b2c3d479:1",
      "input_binding":{
         "Input 1":"VALUE2",
         "Input 2":[
            "VALUE1",
            "VALUE2",
            "VALUE3"
         ],
         "Input 3":null,
         "Input 4":"434"
      }
   }
}
```

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Updated the status successfully.
403	Forbidden	The user attempting to execute this command is not allowed to update the status. In order to prevent this the following check that the following is set: • The user has run privilege rights on this flow. • The user is the owner of the execution or has "manage others" permission. In addition, when performing reassign to another user, the target user must have both `RUN` and `VIEW`privileges on the flow.
409	Conflict	In case the status was already in the requested state.

Execute a Flow by UUID

Request: `POST/executions`

Description: Executes a flow specified by UUID.

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

```
{
       "uuid":"8d52dfc3-1de5-48d4-9c2a-887718de4696",
       “runName”: “run1”,
       “logLevel”: “DEBUG”
       "inputs":
               {
                   "input1":"value for input1",
                   .
                   .
                   .
                   "inputn":"value for inputn"
               },
}
```

`inputs` and `runName`, are optional and can be omitted.

`logLevel` is also optional, the default log level is INFO.

The logLevel attribute receives one of these values: `DEBUG`, `INFO`, and `ERROR`.

Response status codes:

Code	Meaning	Returned When
201	Created	A new flow execution was created.
400	Bad Request	

Response entity body:

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

```
{
	“feedUrl”: “http://localhost:8080/executions/78bec456-db6a-4c05-99ad-0675b230b
feb”
	“executionId”: “78bec456-db6a-4c05-99ad-0675b230bfeb”,
	"errorCode": "value",

}
```

The `feedUrl` is the link to the execution’s status feed. It contains the host name or IP address and not `localhost`. In this example, the action was executed on the local Central server.

In addition, a location header containing a URI to retrieve the created execution for example:

```
/executions/ 78bec456-db6a-4c05-99ad-0675b230bfeb
```

See the Retrieve Feed Events API for more information.

Ad-hoc Flow Execution

Request: `POST/executions`

Description: Ad-hoc Flow execution enables executing a flow without the need to first deploy the flow by providing the AFL flow xml.

Note: This feature will work only when authentication is not enabled in Central.

Note: You can view the afl_xml_schema.xsd file, which describes the HP OO flow AFL (Automation Flow Language). You can view this file in the online help, or download it from **<online-help folder>/content/REST_API_Guide/**. You can also download this file from HPLN, located in the Operations Orchestration 10.x folder in the Resources Tab.

Request entity body:

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

```
{
	"aflContent":"AFL Flow",
	“runName”: “run1”,
	“logLevel”: “DEBUG”
	"inputs":
		{
```

```
                    "input1":"value for input1",
                    .
                    .
                    .
                    "inputn":"value for inputn"
                },
}
```

The `aflContent` must include a JSON encoded AFL flow.

`inputs` and `runName`, are optional and can be omitted.

`logLevel` is also optional, the default log level is `INFO`.

The `logLevel` attribute receives one of these values: `DEBUG`, `INFO`, and `ERROR`.

Response status codes:

Code	Meaning	Returned When
201	Created	A new flow execution was created.
400	Bad Request	

Response entity body:

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

```
{
       "errorCode": "value",
       “executionId”: “78bec456-db6a-4c05-99ad-0675b230bfeb”,
       “feedUrl”: “http://localhost:8080/executions/78bec456-db6a-4c05-99ad-0675b230bf
eb”
}
```

The `feedUrl` is the link to the execution’s status feed. See Get Flow Execution Status for more information.

In addition, a location header containing a URI to retrieve the created execution for example:

```
/executions/ 78bec456-db6a-4c05-99ad-0675b230bfeb
```

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.

Status	Description
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

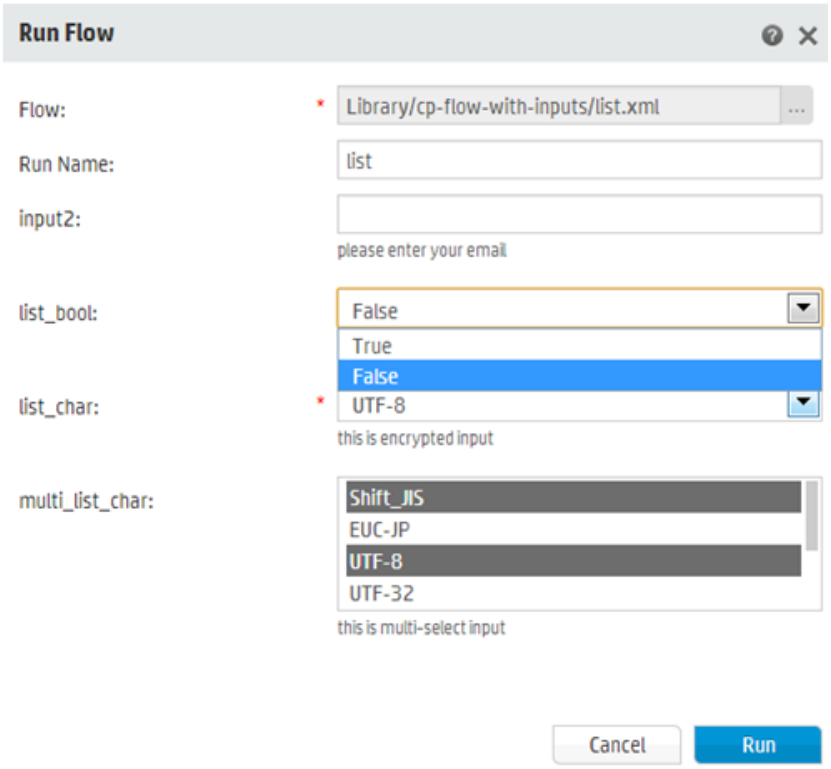
Defines for the client how an input should be presented to the end user.

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

Attribute	Type	Description	Comments
`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
   }
]
```

Flow Library

APIs relating to the Flow Library

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 path variables:

Attribute	Description	Required
`path`	The path that you want to get all tree items under it. Not required. Default value is “”.	Yes

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 Partial Tree

Request: `Get/flows/tree/sub`

Description: Returns a sub tree starting from path and ending in nodePath

Request path variables:

Attribute	Description	Required
`startPath`	The library path which you want to start to search from.	Yes
`nodePath`	The library end path which you want to search to.	

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested tree was found

Response entity body:

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

```
{
   "id":"Library",
   "name":"Library",
   "leaf":false,
   "path":"Library",
   "runnable":false,
   "children":[
      {
         "id":"library/Accelerator Packs",
         "name":"Accelerator Packs",
```

```
            "leaf":false,
            "path":"Library/Accelerator Packs",
            "runnable":false,
            "children":null
         },
         {
            "id":"library/Templates",
            "name":"Templates",
            "leaf":false,
            "path":"Library/Templates",
            "runnable":false,
            "children":[
               {
                  "id":"library/templates/Deprecated",
                  "name":"Deprecated",
                  "leaf":false,
                  "path":"Library/Templates/Deprecated",
                  "runnable":false,
                  "children":null
               },
               {
                  "id":"77a0d53c-c9c0-4f72-922f-d121659d595b",
                  "name":"Check for Windows Event",
                  "leaf":true,
                  "path":"Library/Templates/Check for Windows Event.xml",
                  "runnable":true,
                  "children":null
               }
            ]
         }
      ]
}
```


Find Tree Item By Path

Request: `GET/flows/tree`

Description: Return all tree Items that their name contains search text

Request path variables:

Attribute	Description	Required
`startPath`	The library path which you want to start to search from.	Yes
`nodePath`	The library end path which you want to search to.	Yes
`pageSize`	The page size. Default value is 150.	Yes
`pageNum`	The page number. Default value is 0.	

Response status codes:

Code	Meaning	Returned When
200	OK	The requested items were found

Response entity body:

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

```
[
   {
      "id":"422b8799-c083-4e14-8c92-ab221941ab56",
      "name":"Ping",
      "leaf":true,
      "path":"Library/Operations/Operating Systems/Solaris/Network Operations/Pi
ng.xml",
      "runnable":false,
      "children":null
   },
   {
      "id":"3d1bb4f9-feaf-42aa-85a6-365b502c0a2d",
      "name":"Ping",
      "leaf":true,
      "path":"Library/Operations/Operating Systems/Linux/SUSE Linux/Network Oper
ations/Ping.xml",
      "runnable":false,
      "children":null
   }
]
```

Note: The maximum page size is 150. This will change in future versions.

Get Flow Details

Request: `GET/flows/{uuid}`

Description: Returns flow properties by the uuid.

Request path variables:

Attribute	Description	Required
`uuid`	The flow uuid	Yes

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested flow was found.

Code	Meaning	Returned When
404	Not Found	The requested flow wasn't found or 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":"HPOO-oo-base",
   "version":"version111"
}
```

Get Flow Inputs

Request: `GET/flows/{uuid}/inputs`

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

Request path variables:

Attribute	Description	Required
`uuid`	The flow uuid	Yes

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested flow's inputs were found
404	Not Found	The requested flow wasn't found.

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"
     }
  ]
```

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.

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":" 1381302669536",
       runLogLevel: "DEBUG",
       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":"1328087559000",
       "username":"DavisJ",
       "numOfOccurrences":5,
       runLogLevel: "DEBUG",
       timeZone: "Asia/Amman",
       "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	
403	Forbidden	The user does not have Manage Schedules permission.

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":"1328087559000",
       "username":"DavisJ",
```

```
        "numOfOccurrences":5,
        runLogLevel: "DEBUG",
        timeZone: "Asia/Amman",
        "nextFireTime":null,
        "prevFireTime":null,
        "enabled":false,

        "inputs":
                {
                        "input1":"value for input1",
                        ..."
                        inputn":"value for inputn"
                }
}
```

In addition, a location header containing a URI to retrieve the created schedule for example: `/schedules/1347298851037`

```
{
   "id":"1371112860766",
   "flowScheduleName":"diagnosedDummyFlow",
   "flowUuid":"45647d72-bab4-4e24-bfd8-8c9d00e9cf61",
   "triggerExpression":"*/3600000",
   "startDate":1371112800000,
   "endDate":null,
   "numOfOccurrences":10,
   "timeZone":"Asia/Amman",
   "username":null,
   "runLogLevel":"DEBUG",
   "nextFireTime":null,
   "prevFireTime":null,
   "enabled":false,
   "inputs":{

   }
}
```

Enable Flow-Schedule

Request: `PUT/schedules/{ids}/enabled`

Description: Enable or disable existing flow-schedules.

Request path variables:

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

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.

Response status codes:

Code	Meaning	Returned When
200	OK	The flow-schedules were updated successfully.
403	Forbidden	The user does not have Manage Schedules permissions.

Delete Flow-Schedule

Request: `DELETE/schedules/{ids}`

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

Request path variables:

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

Response status codes:

Code	Meaning	Returned When
200	OK	The flow-schedules were deleted successfully.
400	Bad Request	
403	Forbidden	The user does not have Manage Schedules permissions.

Response entity body:

- **on success:** Returns a JSON string of the id of the schedule that was deleted

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.

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":0,
              "prevFireTime":0,
              "flowScheduleName":"schedule 1",
              "flowName":"flow1",
              "flowPath":"path0",
              "triggerExpression":"0 10 10 ? * 6"
      },
      .
      .
      .
      {
              "id":"567",
              "enabled":true,
              "flowUuid":"3d32e475g-ab54-fe21-df32-4743346ebebd",
              "nextFireTime":0,
              "prevFireTime":0,
              "flowScheduleName":"schedule n",
              "flowName":"flow3",
              "flowPath":"path2",
              "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	Required
`id`	The identifier of the flow-schedule to retrieve.	Yes

Response status codes:

Code	Meaning	Returned When
200	OK	The requested flow-schedule was found.

Code	Meaning	Returned When
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":"253536335",
       "flowScheduleName":"Scheduled Flow Created By REST",
       "flowUuid":"c34de7d6-14cc-4a1c-b25e-85afbb064359",
       "triggerExpression":"0 10 10 ? * 6",
       "startDate":1376072040000,
       "endDate":1377334800000,
       "inputs":
              {
                  "input1":"value for input1",
                  "input2":"value for input2",
                  .
                  .
                  .
                  "inputn":"value for inputn"
              }
}
```

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	Required
id	The identifier of the flow-schedule to update.	Yes

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,
       "inputs":
```

```
                {
                    "input1":"value for input1",
                    .
                    .
                    .
                    "inputn":"value for inputn"
                }
}
```

Response status codes:

Code	Meaning	Returned When
200	OK	The requested flow-schedule was updated 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 path variables:

Attribute	Description	Required
`measurements`	Which statistics to display. If nothing is set then the four statistics are displayed. The following options are available: `roi`, `numOfExecutions`, `avgExecutionTime`, `resultDistribution`.	No

Attribute	Description	Required
sortBy	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).	No
sortDescending	Default is descending.	No
endedBefore	Default is now.	No
endedAfter	Default is one week ago.	No

Response status codes:

Code	Meaning	Returned When
200	OK	Operation was successful
400	Bad Request	• 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.

Response entity body:

List<FlowStatisticsDataVO>:

```
[
  {
   "flowUuid" : 7f2f68-ef48-4a-4a91-bb11-5fadf44bebee",
   "flowPath" : "Library/Repo9x/Dev/ROI/ROI - Basic.xml"
   "flowRoi" : 30.0,
   "numberOfExecutions" : 3
   "averageExecutionTime" : 11000,
   "resultsDistribution" : [
     {
      "type" : "RESOLVED",
      "amount" : 3
     }
    ]
  },
  {
    "flowUuid" : "0b6d9c00-2a34-4c63-8534-4a364d64272",
    "flowPath" : "Library/Repo9x/Dev/ROI/ROI - subflow.xml",
```

```
    "flowRoi" : 1830.0,
    "numberOfExecution" : 3,
    "averageExecutionTime" : 1000,
    "resultsDistribution" : [
      {
       "type" : "SYSTEM_FAILURE",
       "amount" : 2
      },
      {
       "type" : "RESOLVED",
       "amount" : 3
      }
     ]
   }
]
```

Deployment

Deploy Content Packs

Request: `PUT/content-packs/{name:.+}`

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

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

The body of the request should contain the contents of the content pack file to be deployed.

Response entity body:

- **on success:** Returns a JSON value: true

```
{
"aggregatedSeverity":"Info",
"contentPackResponses":{"content pack file name.jar":
   {"contentPackName":" "content pack file"
   name.jar","message":" content pack file name.jar (author: ,
   date:)","responses":[{"contentPackName":" content pack file name.jar","resp
onseCategory":"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" : {
       "contentPackName":"cp.jar",
       "message":"cp.jar (author: , date: )",
        "responses":[
     {
      "contentPackName":"cp.jar",
      "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:
   a. "status":"RUNNING" – with data about the progress
   b. "status":"FINISHED" – with data about the result

Retrieve Details for Deployed Content Packs

Request: `GET/content-packs`

Description: Retrieves details for all deployed content packs.

Response entity body:

An array which contains all current deployed content packs. Each element in the array represents a single content pack.

The returned array is not ordered and appears empty if no content packs exist.

Attribute	Type	Description
`name`	String	The name of the content pack, as it shown in the cp.properties
`version`	String	The version of the content pack, as it shown in the cp.properties

Example:

```
[{
       "name":"base-cp",
       "version":"1.0.0"
}]
```

Response status codes:

Code	Meaning	Returned When
200	OK	All deployed content packs metadata returned.
403	Forbidden	The user does not have the **cpRead** permission

Roll Back Content Packs

Request: `DELETE/content-packs/last`

Description: Roll back the last content pack deployment.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The last deployed content pack had been removed
403	Forbidden	The user does not have Manage Content Packs permission.

Response entity body:

- **on success:** : Returns a Boolean value with that indicates whether the last content pack was removed or not. False indicates that the last content pack was already removed in the past. Removing more than once in a row is not legal.

Create Deployments

Request: `POST/deployments`

Description: Create a deployment process object and return the id to the client.

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: Upload the deployment file and save it into DB for specific deployment process.

Response status codes:

Code	Meaning	Returned When
200	OK	File was uploaded and saved into DB for specific deployment process object.
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.

JSON response example:

```
{
	"files":
	[
		{
			"name":"oo10-business-applications-cp-1.0.117.jar",
			"size":11718432,
			"fileUploadId":101700001,
			"deleteUrl":"/deployments/100900001/files/101700001",
			"deleteType":"DELETE"
		}
	]
}
```

Run a Specific Deployment Process

Request: `PUT/deployments/{deploymentProcessId}s`

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

Response status codes:

Code	Meaning	Returned When
200	OK	The specific deployment process has been started.
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	Another deployment is in process.

Get the Deployment Process Object

Request: `GET/deployments/{deploymentProcessId}`

Description: Get the deployment process object.

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.

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


```
{
       "deploymentProcessId":100900003,
       "status":"RUNNING",
       "currentStep":"Files Downloading",
       "currentStepIndex":1,
       "numOfSteps":15,
       "numOfSubSteps":1,
       "currentSubStep":1,
       "deploymentResultVO":null
}
```


If your deployment process has finished or failed, the JSON appears as follows:


```
{
       "deploymentProcessId":100900004,
       "status":"FINISHED",
       "currentStep":"Commit Data",
       "currentStepIndex":15,
       "numOfSteps":15,
       "numOfSubSteps":1,
       "currentSubStep":0,
       "deploymentResultVO":{
              "aggregatedSeverity":"Info",
              "contentPackResponses":{
                     "cp-config-items.jar":{
                            "contentPackName":"cp-config-tems.jar",
                            "message":"cp-config-items.jar (author: , date: )",
                            "responses":[{
                                   "contentPackName":"cp-config-items.jar",
                                   "responseCategory":"Success",
                                   "level":"Info","message":"Successfully deployed cp-config-items.jar"
                            }]
                     }
              }
```

```
            }
    }
```

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.

Content Configurations

These APIs enable you to manage content configurations, such as create, delete, and update the content configuration.

Create Content Configuration

Request: `POST/content-config`

Description: Creates a content configuration according to the specified `key`, `type` and `value`.

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

```
{
   "value":"value1",
   "key":"mykey1",
   "type":"SYSTEM_PROPERTY"
}
```

Request path parameters:

Parameter	Description	Required
`key`	The name which identifies the content configuration.	Yes
`type`	The type of the content configuration. Valid values: `SELECTION_LIST, SYSTEM_PROPERTY, DOMAIN_TERM, SYSTEM_ACCOUNTS`	Yes
`value`	The value of the content configuration. If the type is set to system account, then inlucde in the value both the name and password.	No

Response status codes:

Code	Meaning	Returned When
201	Created	A content configuration was created successfully.
409	Conflict	A content configuration with the specified `key` already exists.
400	Bad Request	The request body was null.

Response entity body:

- Returns a JSON object of the created content configuration with the following format:

```
{
   "id": "1212417",
   "key":"myKey1",
```

```
   "type":"SYSTEM_PROPERTY",
   "value":"value1"
}
```

- In addition, you get the location header containing the URI to retrieve the created content configuration. For example, `/content-config/myKey1`

Delete Content Configuration

Request: `DELETE/content-config/{name:.+}?type={type}`

Description: Deletes a content configuration according to the specified `key` and `type`.

Request path variables:

Attribute	Description	Required
`name`	The identifier of the content configuration to delete.	Yes

Request path parameters:

Parameter	Description	Required
`type`	The type of the content configuration. Valid values: `SELECTION_LIST, SYSTEM_PROPERTY, DOMAIN_TERM`	Yes

Response status codes:

Code	Meaning	Returned When
200	OK	The content configuration was deleted successfully.
404	Not Found	The requested content configuration was not found.

Get All Content Configurations

Request: `GET/content-config`

Description: Retrieves all the content configurations.

Response status codes:

Code	Meaning	Returned When
200	OK	The requested content configurations were found.
404	Not Found	The requested content configurations were not found.

Response entity body:

- Returns a JSON object with the following format:

```
{
   "contentConfigItemList":
   [
      {
         "id":123,
         "key":"my.key1",
         "type":"SYSTEM_PROPERTY",
         "value":"value1"
      },
      {
         "id":456,
         "key":"my.key2",
         "type":"SYSTEM_PROPERTY",
         "value":"value2"
      }
   ]
}
```

Get Content Configuration

Request: `GET/content-config/{name:.+}?type={type}`

Description: Retrieves the content configuration identified by the specified `key` name.

Request path variables:

Attribute	Description	Required
key	The name which identifies the requested content configuration.	Yes

Request path parameters:

Parameter	Description	Required
type	The type of the content configuration. Valid values: `SELECTION_LIST, SYSTEM_PROPERTY, DOMAIN_TERM`	No

Response status codes:

Code	Meaning	Returned When
200	OK	The requested content configuration was found.
404	Not Found	The requested content configuration was not found.

Response entity body:

- Returns a JSON object with the following format:

```
[{
   "id":123,
   "key":"my.key1",
   "type":"SYSTEM_PROPERTY",
   "value":"value1"
}]
```

Update Content Configuration

Request: `PUT/content-config/{name:.+}?type={type}`

Description: Updates the content configuration identified by the specified `key` name and `type`.

Request path variables:

Attribute	Description	Required
`name`	The name which identifies the content configuration to update.	Yes

Request entity body: The body of this request represents the new value of the content configuration. Write just the value, without wrapping it in JSON.

Request path parameters:

Parameter	Description	Required
`type`	The type of the content configuration. Valid values: `SELECTION_LIST, SYSTEM_PROPERTY, DOMAIN_TERM`	Yes

Response status codes:

Code	Meaning	Returned When
200	OK	The content configuration was updated successfully.
400	Bad Request	The content configuration was not found.

Group Aliases

In addition to Workers groups, there is a further flexibility option to separate the authoring time definition of the group from the runtime definition. In previous versions, the author of a flow was exposed to the runtime topology when a Worker was defined for a specific step in the flow. In this situation, the hostname of the runtime Worker could not change without changing it in all the flows that used it, or it had to be overridden at runtime. See the Concepts Guide for more information.

Create a Workers Group Alias

Request: `POST/group-aliases`

Description: Create a Group Alias for an existing Workers Group.

Request entity body:

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

```
{
"name":"alias name",
"groupName":"associated RASes Group name"
}
```

Where `"name"` is the new Group Alias name and "`groupName`" is the name of the associated Workers Group.

Response status codes:

Code	Meaning	Returned When
201	Created	A new Group Alias was created successfully.
400	Bad Request	The name attribute is missing.
403	Forbidden	The user doesn't have Manage Configuration Items permission.
404	Not Found	The associated Workers Group is not found.
409	Conflict	A Group Alias with the given name was already exist.

Response entity body:

- **on success:** Returns a JSON object of the created group alias with the following format:

  ```
  {
     "name":"alias name",
     "groupName":"associated RASes Group name"
  }
  ```

  In addition, a location header containing a URI to retrieve the created group alias: `/group-aliases/alias%20name`

Get All Groups Aliases

Request: `GET/group-aliases`

Description: Return a list of Group Aliases according to the given start page, page size, and sort direction ordered by name.

Request parameters:

Attribute	Description	Default Value	Required
`start`	Start page number	0	No

Response status codes:

Code	Meaning	Returned When
200	OK	
403	Forbidden	The user doesn't have View Configuration Items or Manage Configuration Items permission.

Response entity body:

- **on success:** Returns a JSON array of the Group Aliases with the following format:

```
[
   {
      "name":"alias1 name",
      "groupName":"associated group name"
   },
   {
      "name":"alias 2 name",
      "groupName":"associated group2 name"
   }
]
```

Where `"name"` is the Group Alias name and `"groupName"` is the name of the associated Workers Group.

Get Group Alias by Name

Request: `GET/group-aliases/{name}`

Description: Get the group alias with the given name.

Request path variables:

Attribute	Description	Required
`name`	The name of the Group Alias	Yes

Response status codes:

Code	Meaning	Returned When
200	OK	
400	Bad Request	The alias name is negative. If no name is provided, the request becomes `Get all group aliases`.
403	Forbidden	The user doesn't have View Configuration Items or Manage Configuration Items permission.
404	Not Found	The requested group alias is found

Response entity body:

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

```
{
   "name":"alias1 name",
   "groupName":"associated group name"
}
```

Delete Group Aliases

Request: `DELETE/group-aliases/{names}`

Description: Delete aliases according to the given alias name, return number of deleted aliases.

Request path variables:

Attribute	Description	Required
names	The names of the Group Aliases to be deleted	Yes

Response status codes:

Code	Meaning	Returned When
200	OK	
403	Forbidden	The user doesn't have Manage Configuration Items permission.

Response entity body:

- **on success:** Returns a JSON value of the number of Group Aliases which were successfully deleted.

Update a Group Alias

Request: `PUT/group-aliases/{name}`

Description: Update an alias according to the given groupAlias and alias name.

Request path variables:

Attribute	Description	Required
name	The name of the Group Alias to be updated	Yes

Request entity body:

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

```
{
   "name":"new alias name",
   "groupName":"associated RASes Group name"
}
```

Response status codes:

Code	Meaning	Returned When
200	OK	The Group Alias update was successful.
400	Bad Request	The name is missing or is negative or the name is empty.
403	Forbidden	The user doesn't have Manage Configuration Items permission.
404	Not Found	The requested Group Alias or Workers Group not found.
409	Conflict	A Group Alias with the specified name already exists.

Note: 405 is returned instead of 400, this is a known limitation.

Response entity body:

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

```
{
   "name":"alias1 name",
   "groupName":"associated group name"
}
```

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 for more information.

Update a Specific Worker

Request: `PUT/workers/{workerId}`

Description: Update an existing worker.

Request path variables:

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

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	

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 haveView Topology or Manage Topology permission.

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
   },
   {
      "uuid":"4440c50e-79d1-45d2-a8dc-94bc42eb9b1f",
      "installPath":"c:\\jenkins\\workspace\\carmel-demo-deployment\\oo\\worke
r",
      "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
   }
]
```

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 doesn't haveView 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	Required
`name`	The name of the Workers group to add	Yes
`WorkersUuids`	The workersUuids of the Worker(s) to be added to the group	Yes

Response status codes:

Code	Meaning	Returned When
200	OK	
400	Bad Request	The name is missing or is negative.
403	Forbidden	The user doesn't have Manage Topology permission.
404	Not Found	The requested group is not found.

Note: 404 is returned instead of 400, this is a known limitation.

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	Required
`name`	The name of the Workers Group to remove from	Yes
`workersUuids`	The uuid of the Worker(s) to remove from the group	Yes

Response status codes:

Code	Meaning	Returned When
200	OK	
400	Bad Request	The name is missing or is negative.
403	Forbidden	The user doesn't have Manage Topology permission.
404	Not Found	The requested group is not found.

Note: 404 is returned instead of 400, this is a known limitation.

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 if 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.

Create a New LDAP Configuration

Request: `POST/authns/ldap-config`

Description: Add a new LDAP configuration.

Request entity body:

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

JSON for an LDAP configuration with both optional and mandatory fields:

```
{
   "hosts":[
      "host_ip_address"
   ],
   "ports":[
      "636"
   ],
   "securedChannel":"true",
   "groupsSearchRecursive":"true",
   "groupsFilter":"(uniqueMember={0})",
   "groupsDns":[
      "ou=groups,dc=devlab,dc=ad"
   ],
   "groupNameAttribute":"cn",
   "userCommonNameAttribute":"cn",
   "usersFilter":"(&amp;(objectclass=person)(uid={0}))",
   "usersDns":[
      "ou=people,dc=devlab,dc=ad"
   ],
   "userIdAttribute":"uid",
   "usersSearchRecursive":"true",
   "privilegedUserDn":"uid=user,ou=people,dc=devlab,dc=ad",
   "privilegedUserPassword":"user_password"
}
```

Note: It is recommended to perform a test for both groups and user configurations before enabling the authentication mode.

`securedChannel`, `groupsSearchRecursive`, `usersSearchRecursive` are all optional. The default value is false.

If `privilegedUserDn`, `privilegedUserPassword` are not provided, an anonymous connection attempt occurs. `userCommonNameAttribute` is optional.

Response status codes:

Code	Meaning	Returned When
201	Successful (Created)	An LDAP configuration was created successfully.
400	Bad Request	
403	Forbidden	The user doesn't have Manage Security Configuration permission.

Response entity body:

- **on success:** Returns a JSON object of the created LDAP configurations with the following format:

```
{
   "hosts":[
      "host_ip_address"
   ],
   "ports":[
      "636"
   ],
   "groupsDns":[
      "ou=groups,dc=devlab,dc=ad"
   ],
   "groupsSearchRecursive":"true",
   "groupsFilter":"(uniqueMember={0})",
   "groupNameAttribute":"cn",
   "usersDns":[
      "ou=people,dc=devlab,dc=ad"
   ],
   "usersSearchRecursive":"true",
   "usersFilter":"(&amp;(objectclass=person)(uid={0}))",
   "userIdAttribute":"uid",
   "userCommonNameAttribute":"cn",
   "userEmailAttribute":"null",
   "securedChannel":"true",
   "privilegedUserDn":"uid=user,ou=people,dc=devlab,dc=ad",
   "privilegedUserPassword":"********",
   "ldapId":"aabf2d25-6b67-4976-8514-3c3f2c3279a8"
}
```


In addition, a location header containing a URI to retrieve the created LDAP configuration for example:

```
/authns/ldap-config/aabf2d25-6b67-4976-8514-3c3f2c3279a8
```

Update an Existing LDAP Configuration

Request: `PUT/authns/ldap-config/{id}`

Description: Update an existing LDAP configuration

Request path variables:

Attribute	Description	Required
`id`	The identifier of the LDAP configuration to update.	Yes

Request entity body:

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

JSON for an LDAP configuration with both optional and mandatory fields:


```
{
   "hosts":[
```

```
      "host_ip_address"
   ],
   "ports":[
      "636"
   ],
   "securedChannel":"true",
   "groupsSearchRecursive":"true",
   "groupsFilter":"(uniqueMember={0})",
   "groupsDns":[
      "ou=groups,dc=devlab,dc=ad"
   ],
   "groupNameAttribute":"cn",
   "userCommonNameAttribute":"cn",
   "usersFilter":"(&amp;(objectclass=person)(uid={0}))",
   "usersDns":[
      "ou=people,dc=devlab,dc=ad"
   ],
   "userIdAttribute":"uid",
   "usersSearchRecursive":"true",
   "privilegedUserDn":"uid=eroth,ou=people,dc=devlab,dc=ad",
   "privilegedUserPassword":"user_password"
}
```

Note: It is recommended to perform a test for both groups and user configurations before enabling the authentication mode.

`securedChannel`, `groupsSearchRecursive`, `usersSearchRecursive` are all optional. The default value is false.

If `privilegedUserDn`, `privilegedUserPassword` are not provided, an anonymous connection attempt occurs. `userCommonNameAttribute` is optional.

Response status codes:

Code	Meaning	Returned When
201	Successful (Created)	An LDAP configuration was created successfully.
400	Bad Request	
403	Forbidden	The user doesn't have Manage Security Configuration permission.

Response entity body:

- **on success:** Returns a JSON object of the updated LDAP configurations.

```
{
   "hosts":[
```

```
      "host_ip_address"
   ],
   "ports":[
      "636"
   ],
   "groupsDns":[
      "ou=groups,dc=devlab,dc=ad"
   ],
   "groupsSearchRecursive":true,
   "groupsFilter":"(uniqueMember={0})",
   "groupNameAttribute":"cn",
   "usersDns":[
      "ou=people,dc=devlab,dc=ad"
   ],
   "usersSearchRecursive":true,
   "usersFilter":"(&amp;(objectclass=person)(uid={0}))",
   "userIdAttribute":"uid",
   "userCommonNameAttribute":"cn",
   "userEmailAttribute":"null",
   "securedChannel":true,
   "privilegedUserDn":"uid=user,ou=people,dc=devlab,dc=ad",
   "privilegedUserPassword":"********",
   "ldapId":"aabf2d25-6b67-4976-8514-3c3f2c3279a8"
}
```


In addition, a location header containing a URI to retrieve the created LDAP configuration for example:

```
/authns/ldap-config/aabf2d25-6b67-4976-8514-3c3f2c3279a8
```

Delete an LDAP Configuration

Request: `DELETE/authns/ldap-config/{id}`

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

Request path variables:

Attribute	Description	Required
`ids`	The identifiers of the LDAP configuration to delete.	Yes

Response status codes:

Code	Meaning	Returned When
204	Successful (no-content)	The configuration was deleted successfully.

Code	Meaning	Returned When
403	Forbidden	The user doesn't have Manage Security Configuration permission.
404	Not found	

Get LDAP Configurations

Request: `GET/authns/ldap-config/{ldapId}`

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

Request path variables:

Attribute	Description	Required
`ldapId`	The id of the required LDAP configurations.	Yes

Response status codes:

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

Response entity body:

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

```
{
   "hosts":[
      "ip_address"
   ],
   "ports":[
      636
   ],
   "groupsDns":[
      "ou=groups,dc=devlab,dc=ad"
   ],
   "groupsSearchRecursive":true,
   "groupsFilter":"(uniqueMember={0})",
   "groupNameAttribute":"cn",
   "usersDns":[
      "ou=people,dc=devlab,dc=ad"
   ],
```

```
   "usersSearchRecursive":true,
   "usersFilter":"(&amp;(objectclass=person)(uid={0}))",
   "userIdAttribute":"uid",
   "userCommonNameAttribute":"cn",
   "userEmailAttribute":null,
   "securedChannel":true,
   "privilegedUserDn":"uid=user,ou=people,dc=devlab,dc=ad",
   "privilegedUserPassword":"********",
   "ldapId":"d0c76e23-9a89-471c-b8d3-0441ede87595"
}
```

The `privilegedUserPassword` is returned with asterisks if a password exists.

Test User Attributes

Request: `POST/authns/ldap-users`

Description: Retrieves the request user's LDAP attributes with the given LDAP configurations.

Request entity body:

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

JSON for an LDAP user test with both optional and mandatory fields:

```
{
   "groupNameAttribute":"cn",
   "groupsDns":[
      "ou=groups,dc=devlab,dc=ad"
   ],
   "groupsFilter":"(uniqueMember={0})",
   "groupsSearchRecursive":true,
   "hosts":[
      "host_ip_address"
   ],
   "ports":[
      "636"
   ],
   "privilegedUserDn":"uid=user,ou=people,dc=devlab,dc=ad",
   "privilegedUserPassword":"user_password",
   "securedChannel":"true",
   "userId":"user",
   "userIdAttribute":"uid",
   "usersDns":[
      "ou=people,dc=devlab,dc=ad"
   ],
   "usersFilter":"(&amp;(objectclass=person)(uid={0}))"
}
```

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Succeeded in retrieving user attributes from the LDAP.
400	Bad Request	
403	Forbidden	The user doesn't have Manage Security Configuration permission.

Response entity body:

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


```
{
   "userId":"user",
   "commonName":"common_name",
   "emails":"null",
   "attributes":[
      {
         "name":"objectClass",
         "value":"person"
      },
      {
         "name":"givenName",
         "value":"firstname"
      },
      {
         "name":"uid",
         "value":"user"
      },
      {
         "name":"cn",
         "value":"user"
      },
      {
         "name":"sn",
         "value":"surname"
      },
      {
         "name":"userPassword",
         "value":"{SSHA}Hj9tpIqw1UziuDViCdQaFzK/+ccKTbmlQbelDQ=="
      },
      {
         "name":"mail",
         "value":"myemail@mail.com"
      }
   ]
}
```


The attribute contains all the retrieve LDAP attributes, and may vary.

- **on bad request:** Returns a JSON object with the following format:

```
{
   "message":"The entry ou=people,dc=devlab,dc=a specified as the search base
does not exist in the Directory Server"
}
```

Test User Groups

Request: `POST/authns/ldap-groups`

Description: Retrieves the request user's groups names with the given LDAP configurations.

Request entity body:

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

JSON for an LDAP user groups test with both optional and mandatory fields:

```
{
   "groupNameAttribute":"cn",
   "hosts":[
      "host_ip_address"
   ],
   "ports":[
      636
   ],
   "privilegedUserDn":"uid=user,ou=people,dc=devlab,dc=ad",
   "privilegedUserPassword":"user",
   "securedChannel":true,
   "userCommonNameAttribute":"cn",
   "userId":"user",
   "userIdAttribute":"uid",
   "usersDns":[
      "ou=people,dc=devlab,dc=ad"
   ],
   "usersFilter":"(&amp;(objectclass=person)(uid={0}))",
   "usersSearchRecursive":true
}
```

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Succeeded in retrieving user groups names from the LDAP.
400	Bad Request	
403	Forbidden	The user doesn't have Manage Security Configuration permission.

Response entity body:

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

```
[
   {
      "groupName":"oo"
   },
   {
      "groupName":"oo2"
   }
]
```

- **on bad request:** Returns a JSON object with the following format:

```
{
   "message":"The entry ou=groups,dc=devlab,dc=a specified as the search base
does not exist in the Directory Server"
}
```

Test User Attributes with Existing LDAP Password

Request: `POST/authns/ldap-users/{ldapId}`

Description: Retrieves the request user's LDAP attributes with the given LDAP configurations.The password saved for this ldapId can be used.

Request path variables:

Attribute	Description	Required
`ldapId`	The id of the LDAP with a saved privileged user password.	Yes

Request entity body:

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

JSON for an LDAP user test with both optional and mandatory fields:

```
{
   "groupNameAttribute":"cn",
   "hosts":[
      "host_ip_address"
   ],
   "ports":[
      "636"
   ],
   "privilegedUserDn":"uid=user,ou=people,dc=devlab,dc=ad",
   "privilegedUserPassword":"********",
   "securedChannel":true,
   "userCommonNameAttribute":"cn",
   "userId":"user",
   "userIdAttribute":"uid",
```

```
   "usersDns":[
      "ou=people,dc=devlab,dc=ad"
   ],
   "usersFilter":"(&amp;(objectclass=person)(uid={0}))",
   "usersSearchRecursive":true
}
```

If the `privilegedUserPassword` is provided with asterisks, the server will use the existing password. This would be the main motivation behind this request.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Succeeded in retrieving user attributes from the LDAP.
400	Bad Request	
403	Forbidden	The user doesn't have Manage Security Configuration permission.

Response entity body:

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

```
{
   "userId":"user",
   "commonName":"common_name",
   "emails":"null",
   "attributes":[
      {
         "name":"objectClass",
         "value":"person"
      },
      {
         "name":"givenName",
         "value":"firstname"
      },
      {
         "name":"uid",
         "value":"user"
      },
      {
         "name":"cn",
         "value":"user"
      },
      {
         "name":"sn",
         "value":"surname"
      },
      {
         "name":"userPassword",
```

```
            "value":"{SSHA}Hj9tpIqw1UziuDViCdQaFzK/+ccKTbmlQbelDQ=="
         },
         {
            "name":"mail",
            "value":"myemail@mail.com"
         }
      ]
}
```

The attribute contains all the retrieve LDAP attribute, and may vary.

- **on bad request**: Returns a JSON object with the following format:

```
{
   "message":"The entry ou=people,dc=devlab,dc=a specified as the search base
does not exist in the Directory Server"
}
```

Test User Groups with Existing LDAP Password

Request: `POST/authns/ldap-groups/{ldapId}`

Description: Retrieves the request user's groups names with the given LDAP configurations.

The password saved for this ldapId can be used.

Request path variables:

Attribute	Description	Required
`ldapId`	The id of the LDAP with a saved privileged user password.	Yes

Request entity body:

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

JSON for an LDAP user groups test with both optional and mandatory fields:

```
{
   "groupNameAttribute":"cn",
   "groupsDns":[
      "ou=groups,dc=devlab,dc=ad"
   ],
   "groupsFilter":"(uniqueMember={0})",
   "groupsSearchRecursive":true,
   "hosts":[
      "host_ip_address"
   ],
   "ports":[
      "636"
   ],
   "privilegedUserDn":"uid=user,ou=people,dc=devlab,dc=ad",
```

```
   "privilegedUserPassword":"********",
   "securedChannel":true,
   "userId":"eroth",
   "userIdAttribute":"uid",
   "usersDns":[
      "ou=people,dc=devlab,dc=ad"
   ],
   "usersFilter":"(&amp;(objectclass=person)(uid={0}))"
}
```

If the `privilegedUserPassword` is provided with asterisks, the server will use the existing password. This is the initial reason for this request.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Succeeded in retrieving user attributes from the LDAP.
400	Bad Request	
403	Forbidden	The user doesn't have Manage Security Configuration permission.

Response entity body:

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

```
[{"groupName":"oo"},{"groupName":"oo2"}]
```

- **on bad request:** Returns a JSON object with the following format:

```
{
"message":"The entry ou=groups,dc=devlab,dc=a specified as the search base doe
s not exist in the Directory Server"
}
```

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

```
{
   "password":"12345",
   "roles":[
      {
         "name":"EVERYONE"
      },
      {
         "name":"PROMOTER"
      }
   ],
   "username":"mranderso"
}
```

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	
403	Forbidden	The user doesn't have Manage Security Configuration permission.

Response entity body:

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

```
{
   "displayName":"mranderson",
   "userId":"mranderson",
   "emails":null,
   "hasPassword":true,
   "roles":[
      "END_USER",
      "PROMOTER",
      "SYSTEM_ADMIN"
   ],
   "permissions":null
}
```

In addition, a location header containing a URI to retrieve the created user configuration, for example:

```
/users/mranderson
```

Update Existing Role

Request: `PUT/users/{username}`

Description: Update an existing internal user

Request path variables:

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

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": "mranderso"
}
```

Note: The variable `{id}` refers to which user to update, while the user name field refers to the new user name.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Updated the LDAP configurations successfully.
400	Bad Request	
403	Forbidden	The user doesn't have Manage Security Configuration permission.

Response entity body:

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

```
{
   "displayName":"mranderson",
   "userId":"mranderson",
   "emails":null,
   "hasPassword":true,
   "roles":[
      "END_USER",
      "PROMOTER",
      "SYSTEM_ADMIN"
   ],
```

```
   "permissions":null
}
```

Delete an Internal User

Request: `DELETE/users/{userIds}`

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

Request path variables:

Attribute	Description	Required
`userIds`	The identifiers of the internal users to delete.	Yes

Response status codes:

Code	Meaning	Returned When
204	Successful (no-content)	The internal users no longer exist in the system.
403	Forbidden	The user doesn't have Manage Security Configuration permission.

Note: A logged in user cannot delete their own internal user account.

Get Users

Request: `GET/users?origin=internal`

Description: Retrieves users

Request parameters:

Attribute	Description	Default Value	Required
`origin`	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 doesn't 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,
      "hasPassword":true,
      "roles":[
         "ADMIN"
      ],
      "permissions":null
   },
   {
      "displayName":"mranderson",
      "userId":"mranderson",
      "emails":null,
      "hasPassword":true,
      "roles":[
         "END_USER"
      ],
      "permissions":null
   },
   {
      "displayName":"rothjohn",
      "userId":"rothjohn",
      "emails":null,
      "hasPassword":true,
      "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,
   "hasPassword":false,
   "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 session's user was returned.
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:

```
{
   "enabled":true,
   "initString":"CENTRAL_PASSPHRASE",
   "domain":"mydomain1.com",
   "protectedDomains":[
      "mydomain1.com",
      "mydomain2.com"
   ]
}
```

Update LW SSO configuration

Request: `PUT/authns/lwsso-config`

Description: Updates the lightweight SSO configuration.

Request path variables:

Attribute	Description	Required
`id`	The identifier of the internal user to update.	Yes

Request entity body:

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

JSON for a LWSSO User update with both optional and mandatory fields:

```
{
   "enabled":false,
   "initString":"CENTRAL_PASSPHRASE_NEW",
   "domain":"mydomainnew1.com",
   "protectedDomains":[
      "mydomainnew1.com",
      "mydomainnew2.com"
   ]
}
```

Response status codes:

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

Response entity body:

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

```
{
   "enabled":false,
   "initString":"CENTRAL_PASSPHRASE_NEW",
   "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
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:

```
{
   "enable":true
}
```

Update Authentication configurations

Request: `PUT/authns`

Description: Updates the authentication configurations.

Request entity body:

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

JSON for a LWSSO User update with both optional and mandatory fields:

```
{
   "enable":true
}
```

Response status codes:

Code	Meaning	Returned When
204	Successful (no-content)	The authentication configuration were updated.
403	Forbidden	The user doesn't have Manage Security Configuration permission.

Response entity body:

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

```
{
   "enable":true
}
```

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	Required
`roleName`	The name of the required role.	Yes

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested role was found.
403	Forbidden	The user doesn't 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":"ADMIN",
   "description":"Administration Role",
   "permissions":[
      "topologyManage",
      "configurationItemManage",
      "systemSettingsManage",
      "securityConfigRead",
      "flowPermissionManage",
      "scheduleManage",
      "systemSettingsRead",
      "scheduleRead",
      "othersRunsManage",
      "configurationItemRead",
      "topologyRead",
      "cpManage",
      "securityConfigManage",
      "cpRead"
   ],
   "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":"ADMIN",
      "permissions":[
         "securityConfigRead",
         "cpRead",
         "topologyManage",
         "securityConfigManage",
         "configurationItemRead",
         "scheduleManage",
         "topologyRead",
         "othersRunsManage",
         "configurationItemManage",
         "systemSettingsManage",
         "flowPermissionManage",
         "cpManage",
         "scheduleRead",
         "systemSettingsRead"
      ],
      "groupsNames":[

      ],
      "description":"Administration Role"
   },
   {
      "name":"EVERYONE",
      "permissions":[

      ],
      "groupsNames":[

      ],
      "description":"Everyone Role"
   },
   {
      "name":"PROMOTER",
      "permissions":[
         "configurationItemManage",
         "cpRead",
         "configurationItemRead",
         "flowPermissionManage",
         "cpManage"
      ],
      "groupsNames":[

      ],
      "description":"Promoter Role"
   },
   {
```

```
      "name":"SYSTEM_ADMIN",
      "permissions":[
         "securityConfigRead",
         "topologyRead",
         "systemSettingsManage",
         "topologyManage",
         "securityConfigManage",
         "systemSettingsRead"
      ],
      "groupsNames":[

      ],
      "description":"System Administrator Role"
   },
   {
      "name":"END_USER",
      "permissions":[

      ],
      "groupsNames":[

      ],
      "description":"End User Role"
   }
]
```

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":"ADMIN",
      "permissions":[
         "securityConfigRead",
         "cpRead",
         "topologyManage",
         "securityConfigManage",
         "configurationItemRead",
         "scheduleManage",
         "topologyRead",
         "othersRunsManage",
```

```
            "configurationItemManage",
            "systemSettingsManage",
            "flowPermissionManage",
            "cpManage",
            "scheduleRead",
            "systemSettingsRead"
         ],
         "groupsNames":[

         ],
         "description":"Administration Role"
      },
      {
         "name":"EVERYONE",
         "permissions":[

         ],
         "groupsNames":[

         ],
         "description":"Everyone Role"
      },
      {
         "name":"PROMOTER",
         "permissions":[
            "configurationItemManage",
            "cpRead",
            "configurationItemRead",
            "flowPermissionManage",
            "cpManage"
         ],
         "groupsNames":[

         ],
         "description":"Promoter Role"
      },
      {
         "name":"SYSTEM_ADMIN",
         "permissions":[
            "securityConfigRead",
            "topologyRead",
            "systemSettingsManage",
            "topologyManage",
            "securityConfigManage",
            "systemSettingsRead"
         ],
         "groupsNames":[

         ],
```

```
      "description":"System Administrator Role"
   },
   {
      "name":"END_USER",
      "permissions":[

      ],
      "groupsNames":[

      ],
      "description":"End User 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	
403	Forbidden	The user doesn't have Manage Security Configuration permission.

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":[
      "cpRead",
      "topologyManage",
      "systemSettingsManage",
      "securityConfigManage",
      "topologyRead",
      "configurationItemManage",
      "systemSettingsRead",
      "securityConfigRead",
      "scheduleRead",
      "othersRunsManage",
      "flowPermissionManage",
      "cpManage",
      "configurationItemRead",
      "scheduleManage"
```

```
   ],
   "groupsNames":[
      "Super Group"
   ]
}
```

Update an Existing Role

Request: `PUT/roles/{roleName}`

Description: Update an existing role

Request path variables:

Attribute	Description	Required
`roleName`	The name of the role to update.	Yes

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	
403	Forbidden	The user doesn't have Manage Security Configuration permission.

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	Required
`roleName`	The identifier of the role name to delete.	Yes

Response status codes:

Code	Meaning	Returned When
204	Successful (no-content)	The role was deleted successfully.
403	Forbidden	The user doesn't 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.

Code	Meaning	Returned When
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:

```
{"defaultRole":"EVERYONE"}
```

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

Update the Default Role

Request: `PUT/roles/default-name`

Description: Update an existing role.

Request entity body:

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

JSON for a default role update with both optional and mandatory fields:

```
{defaultRole:PROMOTER}
```

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Updated the default role successfully.
403	Forbidden	The user doesn't have Manage Security Configuration permission.

Response entity body:

- **on success:** Returns a JSON object of the updated default role.

```
{
   "defaultRole":"PROMOTER"
}
```

Get Entitlements Per Path and Roles

Request: `GET/roles/{rolesNames}/entitlements/**`

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

Example:

```
roles/ADMIN%2CEVERYONE%2CPROMOTER%2CSYSTEM_ADMIN%2CEND_USER/entitlements/Library
/cp-parallel
```

Note: %2C is encoded comma.

Request path variables:

Attribute	Description	Required
rolesNames	The roles for which the entitlements are requested. **Note**: The entitlement path.	Yes

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	The requested entitlements was found.
400	Bad request	
403	Forbidden	The user doesn't have Manage Flow Permissions permission.
404	Not Found	The requested role was not found.

Response entity body:

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

```
{
   "ADMIN":[
      "RUN",
      "VIEW"
   ],
   "EVERYONE":[

   ],
   "END_USER":[

   ],
   "SYSTEM_ADMIN":[

   ],
   "PROMOTER":[
      "RUN",
      "VIEW"
   ]
}
```

Update Path Entitlement Per Role

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

Description: Update specific entitlement path per role.

Example:

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

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 is Recursive is false.

Response status codes:

Code	Meaning	Returned When
200	Successful (OK)	Updated the role successfully.
400	Bad Request	
403	Forbidden	The user doesn't have Manage Flow Permissions permission.

Response entity body:

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

```
{
   "privileges":[
      "VIEW",
      "RUN"
   ],
   "isRecursive":true
}
```

Configuration Items

These APIs enable you to manage the system configuration.

Create a Configuration Item

Request: `POST/config`

Description: Creates a 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 configuration item was created successfully.
403	Forbidden	The user doesn't have Manage System Settings permission.

Response entity body:

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

```
{
       “id”: “1179648”,
       "key":"myKey",
       "value":"value"
}
```

In addition, a location header containing a URI to retrieve the created configuration item: `/config/myKey`

Get All Configuration Items

Request: `GET/config`

Description: Retrieves all configuration items.

Response status codes:

Code	Meaning	Returned When
200	OK	All existing configuration items were retrieved.

Code	Meaning	Returned When
403	Forbidden	The user doesn't have View System Settings or Manage System Settings permission.
404	Not Found	No configuration items were retrieved.

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 Configuration Item

Request: `GET/config/{key:.+}`

Description: Retrieves a configuration item by key.

Request path variables:

Attribute	Description	Required
key	The key of the requested configuration item.	Yes

Response status codes:

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

Response entity body:

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

Update Configuration Item

Request: `PUT/config/{key:.+}`

Description: Updates an existing configuration item.

Request path variables:

Attribute	Description	Required
`key`	The key of the configuration item to update.	Yes

Request entity body: The body of this request needs to include a JSON string with the new value of the configuration item.

Response status codes:

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

Response entity body:

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

Miscellaneous

Get HP OO Version

Request: `GET/version`

Description: Retrieves information about the HP OO version.

Response entity body:

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

```
{
   "version":"1.1.1.1",
   "revision":"1.1",
   "build number":"123"
}
```


Java™
COMPATIBLE