

# HP Operations Orchestration

for the Windows and Linux operating systems

Software Version: OO Content Pack 7

---

## HP Universal Configuration Management Database Integration Guide

Document Release Date: January 2012

Software Release Date: January 2012



## 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 Notices

© Copyright <date range: year first released to year of current release; example: 2005-2010> Hewlett-Packard Development Company, L.P.

### Trademark Notices

For information on open-source and third-party software acknowledgements, see *Open-Source and Third-Party Software Acknowledgements* (3rdPartyOpenNotices.pdf) in the documentation set for this release.

## 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 Web site at:

**[www.hp.com/go/hpsoftwaresupport](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](http://h20230.www2.hp.com/new_access_levels.jsp)**

# Contents

- 1 Introduction .....7
  - Purpose of the UCMDB Integration .....8
  - Supported Versions .....8
  - Downloading OO Releases and Documents on HP Live Network .....8
- 2 Getting Started with the UCMDB Integration .....9
  - Installing and Configuring the Integration .....10
  - UCMDB — OO Integration Architecture .....10
  - UCMDB Use Cases .....10
- 3 Using the UCMDB – OO Integration .....12
  - Location of UCMDB Integration Operations and Flows in OO Studio .....13
  - Common Inputs in the Integration .....13
  - Descriptions of UCMDB Integration Operations and Flows .....14
    - Get Context-based Discovery Status .....14
    - Stop Context-based Discovery .....16
    - Trigger Context-based Discovery .....17
    - Add Object .....18
    - Add Relationship .....19
    - Delete Object .....20
    - Delete Relation .....21
    - Get Filtered Object List by Type .....21
    - Get Neighboring Objects .....22
    - Get Object Attributes by Id .....23
    - Get Object List By Host Name .....24
    - Get Object List by Type .....25
    - Get Query Name of View .....25
    - Get Topology Map by Query Name .....26
    - Get Server Status .....26
    - Get Topology Map by Query Name with Parameter .....27
    - Get Topology Map by View Name .....28
    - Is Object Related To .....29
    - Modify Object Attribute(s) .....29
    - Validate IP in UCMDB .....30
- 4 Troubleshooting .....32
  - Troubleshooting Overview .....33

General Troubleshooting Procedures and Tools .....	33
Error Messages .....	36
<b>5 Security .....</b>	<b>39</b>
About UCMDB Security .....	40
<b>6 OO Tools .....</b>	<b>41</b>
OO Tools You Can Use with the UCMDB-OO Integration.....	42

# 1 Introduction

This section includes the following topics:

- [Purpose of the UCMDB Integration](#)
- [Supported Versions](#)
- [Downloading OO Releases and Documents on HP Live Network](#)

## Purpose of the UCMDB Integration

With the HP Universal Configuration Management Database (UCMDB) integration, administrators can build HP Operations Orchestration (OO) flows that are integrated with HP UCMDB.

This document explains how this integration has been implemented and how the integration OO operations communicate between OO and UCMDB.

## Supported Versions

**Table 1 Supported Versions**

Operations Orchestration Version	UCMDB Version
OO Content Pack 7	7.0, 8.0, 9.0x

## Downloading OO Releases and Documents on HP Live Network

HP Live Network provides an **Operations Orchestration Community** page where you can find and download supported releases of OO and associated documents.

To download OO releases and documents, visit the following site:

<https://hpln.hp.com/>

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.

On the **HP Live Network** page, click **Operations Orchestration Community**.

**The Operations Orchestration Community** page contains links to announcements, discussions, downloads, documentation, help, and support.

1. On the left-hand side, click **Operations Orchestration Content Packs**.
2. In the **Operations Orchestration Content Packs** box, click **Content**. The HP Passport and sign-in page appears.
3. Enter your user ID and Password to access to continue.
4. Click **HP Operations Orchestration 9.00**.
5. Search for HP Operations Orchestration Content Pack 7



## 2 Getting Started with the UCMDB Integration

This section includes the following topics:

- [Installing and Configuring the Integration](#)
- [UCMDB – OO Architecture](#)
- [UCMDB Use Cases](#)

## Installing and Configuring the Integration

There is no special integration installation and configuration needed for the HP UCMDB integration. Just make sure that the system that has the RSJRAS service running on it can access the UCMDB server.

The default UCMDB access URL is:

**`http://<ucmdbserver>:8080/ucmdb`**

using the default username admin and the default password admin. However, we recommend creating an integration user on the UCMDB server. To learn how to create an integration user, refer to the UCMDB documentation.

## UCMDB — OO Integration Architecture

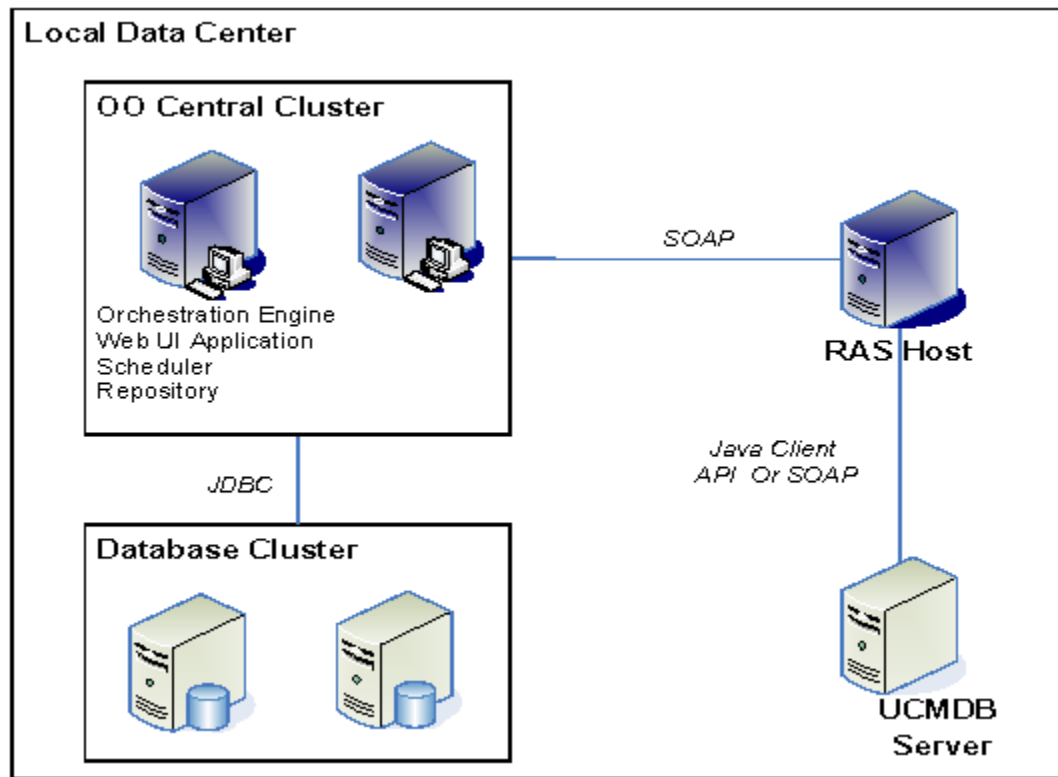


Figure 1 - UCMDB - OO Integration Architecture

## UCMDB Use Cases

Following are the major use cases for the UCMDB integration, and the operations and flows that you can use to implement them.

1 Execute a Topology Query and obtain a Topology Map:

- Get Topology Map by Query Name
- Get Topology Map by Query Name with Parameter
- Get Topology Map by View Name
- Get Query Name of View

2 Manage Configuration Items and relationships:

- Add Object
- Delete Object
- Add Relationship
- Delete Relation
- Get Neighboring Objects
- Get Filtered Object List by Type
- Get Object List by Type
- Get Object Attributes by Id
- Modify Object Attribute(s)

3 UCMDB IP validation:

- Validate IP in UCMDB
- Get Object List By Host Name
- Is Object Related To

4 Trigger/stop focused discovery:

- Trigger Context-based Discovery
- Stop context-based Discovery
- Get Context-based Discovery Status

---

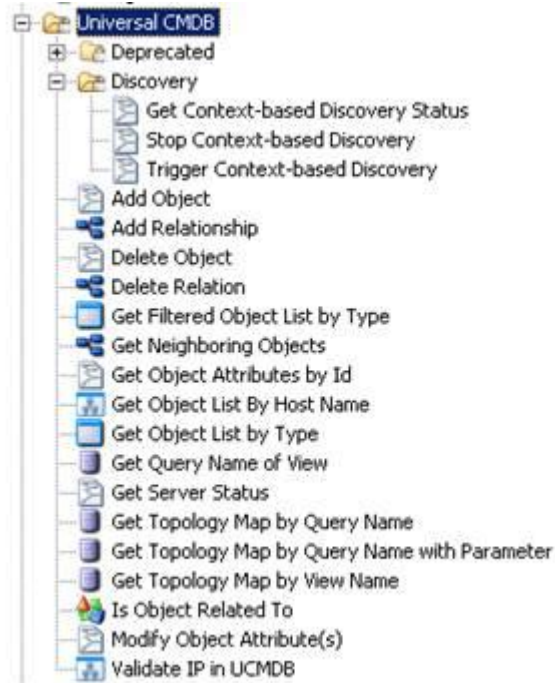
## 3 Using the UCMDB – OO Integration

This section includes the following topics:

- [Location of UCMDB Integration Operations and Flows in OO Studio](#)
- [Common Inputs in the Integration](#)
- [Descriptions of UCMDB Integration Operations and Flows](#)

# Location of UCMDB Integration Operations and Flows in OO Studio

The HP UCMDB integration includes the following operations in the Studio Library, in the Library/Integrations/Hewlett-Packard/Universal CMDB/ folder.



**Figure 2 – Location of UCMDB integration operations and flows**

The operations support the following types of tasks:

- Add, remove, and update Configuration Items in the UCMDB.
- Add, remove, and update Relations in the UCMDB.
- Retrieve information about Configuration Items and Relationships in the UCMDB.
- Validate an IP or a host name in the UCMDB.
- Trigger, stop context-based focused discovery in the UCMDB.

## Common Inputs in the Integration

The following inputs are used consistently throughout the HP UCMDB integration's operations.

**cmdbHost**

The host of the UCMDB server. You can specify it as an IP address or a DNS name.

**cmdbPort**

The port of the UCMDB server on which the CMDB is running. The default port is 8080.

### Protocol

The protocol used to connect to the uCMDB server. Note that https (ssl) is currently only supported by uCMDB 9.x.

Valid values: http, https

Default value: http

### username

The username to connect to the UCMDB server.

### password

The password for the username.

### cmdbVersion

The major version number of UCMDB (for example, 7 for UCMDB 7.0, 8 for UCMDB 8.0, or 9 for UCMDB 9.0). The default is 7.

## Descriptions of UCMDB Integration Operations and Flows

This section describes the UCMDB integration's operations and flows, including any operation- or flow-specific inputs.

### Get Context-based Discovery Status

Retrieve status of the DDM discovery tasks/jobs associated with your specified discovery context. There are three types of discovery context info you can provide: CIS (a list of existing uCMDB CIs), JOBS (a list of existing uCMDB discovery job names), IP\_RANGES (a list of IPs and/or IP ranges).

**Note:** Currently this operation is only available and supported for UCMDB/DDM 9.x.

### Input

All of the operation's inputs except the following are described in [Common Inputs in the Integration](#).

#### contextType

The type of the discovery context.

Valid values: CIS, JOBS, IP\_RANGES

Default value: CIS

#### discoveryContext

The discovery context data of the chosen type.

For type CIS, valid values are one or a list of CI Ids. The Ids should be valid and for existing UCMDB CIs, separated by the delimiter.

Examples: 76784075bfa057ccefd2d6130879cf7f,41a38b4da9a18faeb951cb9f33a13b7e

For type JOBS, valid values are one or a list of discovery job names. The job names should be separated by the delimiter.

Examples: Host Connection by Shell,Databases TCP Ports

For type IP\_RANGES, valid values are one or a list of IP addresses and/or ranges. IP ranges should use ":" between the start address and the end address.

Examples: 15.43.21.5,16.67.10.0:16.67.10.20

### delimiter

The delimiter used between the discoveryContext elements.

Default value:","

## Responses

The operation returns the following results:

### success

The discovery status is successfully retrieved

### failure

The operation failed with thrown exception

## Results

The operation returns the following results:

### returnResult

This is the primary output.

### successCount

The number of successfully completed discovery tasks triggered by the context.

### failedCount

The number of failed discovery tasks triggered by the context.

### pendingCount

The number of pending (unfinished) discovery tasks triggered by the context.

### totalCount

The total number of all the discovery tasks triggered by the context, including successCount, failedCount and pendingCount.

### discoveryErrors

The discovery errors message of the errors occurred during discovery.

### discoveryWarnings

The discovery warnings message of the warnings during discovery.

## Stop Context-based Discovery

Stop execution of the DDM discovery tasks/jobs associated with your specified discovery context. There are three types of discovery context info you can provide: CIS (a list of existing UCMDB CIs), JOBS (a list of existing UCMDB discovery job names), IP\_RANGES (a list of IPs and/or IP ranges).

**Note:** Currently this operation is only available and supported for UCMDB/DDM 9.x.

### Input

All of the operation's inputs except the following are described in [Common Inputs in the Integration](#)

#### contextType

The type of the discovery context.

Valid values: CIS, JOBS, IP\_RANGES

Default value: CIS

#### discoveryContext

The discovery context data of the chosen type.

For type CIS, valid values are one or a list of CI Ids. The Ids should be valid and for existing UCMDB CIs, separated by the delimiter.

Examples: 76784075bfa057ccefd2d6130879cf7f,41a38b4da9a18faeb951cb9f33a13b7e

For type JOBS, valid values are one or a list of discovery job names. The job names should be among the activated discovery jobs in the UCMDB server, separated by the delimiter.

Examples: Host Connection by Shell,Databases TCP Ports

For type IP\_RANGES, valid values should be one or a list of IP addresses and/or ranges. IP ranges should use ":" between the start address and the end address.

Examples: 15.43.21.5,16.67.10.0:16.67.10.20

#### delimiter

The delimiter used between the discoveryContext elements.

Default value: ","

### Responses

The operation returns the following results:

#### success

The discovery tasks/jobs associated with the given context are stopped.

#### failure

The operation failed with thrown exception.

### Results

The operation returns the following result:



`returnResult`

This is the primary output.

## Trigger Context-based Discovery

Trigger DDM discovery tasks/jobs associated with your specified discovery context. There are three types of discovery context info you can provide to trigger DDM discovery: CIS (a list of existing uCMDB CIs), JOBS (a list of existing CMDB discovery job names), IP\_RANGES (a list of IPs and/or IP ranges).

**Note:** Currently this operation is only available and supported for UCMD/DDM 9.x.

### Input

All of the operation's inputs except the following are described in [Common Inputs in the Integration](#)

`contextType`

The type of the discovery context.

Valid values: CIS, JOBS, IP\_RANGES

Default value: CIS

`discoveryContext`

The discovery context data of the chosen type.

For type CIS, valid values are one or a list of CI Ids. The Ids should be valid and for existing UCMD/ CIs, separated by the delimiter.

Examples: 76784075bfa057ccefd2d6130879cf7f,41a38b4da9a18faeb951cb9f33a13b7e

For type JOBS, valid values are one or a list of discovery job names. The job names should be among the activated discovery jobs in the UCMD/ server, separated by the delimiter.

Examples: Host Connection by Shell,Databases TCP Ports

For type IP\_RANGES, valid values are one or a list of IP addresses and/or ranges. IP ranges should use ":" between the start address and the end address.

Examples: 15.43.21.5,16.67.10.0:16.67.10.20

`delimiter`

The delimiter used between the discoveryContext elements.

Default value: ","

`waitTime`

The time in seconds for RAS to wait for all the triggered discovery jobs and tasks to complete or fail, that is, no more pending tasks/jobs.

Default value: 0

### Responses

The operation returns the following results:

### Success

The discovery is triggered and completed without any error, but could have some warnings. See result warningsMsg.

### passWithError

The discovery is triggered and completed with some errors and some discovery tasks failed. See results errorsMsg and failedCount for the error message and the number of failed discovery tasks.

### noDiscoveryJob

No discovery task/job is found in UCMDB that is associated with your specified discovery context. Make sure your discovery context input is valid.

### failure

The operation failed with thrown exception.

## Results

The operation returns the following results:

### returnResult

This is the primary output.

### successCount

The number of successfully completed discovery tasks triggered by the context.

### failedCount

The number of failed discovery tasks triggered by the context.

### pendingCount

The number of pending (unfinished) discovery tasks triggered by the context.

### totalCount

The total number of all the discovery tasks triggered by the context, including successCount, failedCount and pendingCount.

### discoveryErrors

The discovery errors message of the errors occurred during discovery.

### discoveryWarnings

The discovery warnings message of the warnings during discovery.

## Add Object

The **Add Object** operation adds a Configuration Item to the UCMDB that has the key property or other properties specified in the **objectType** input. It returns the ID of the created Configuration Item.

## Inputs

All of the operation's inputs except the following are described in [Common Inputs in the Integration](#).

### objectType

The Configuration Item Type (CIT) of the UCMDB. For example, you can specify an **objectType** of **nt**, which is the class name for the Configuration Item Type with the display name **Windows**. You can find the class name in the CI Type Manager by clicking the **Details** tab for the CI item. The **Name** field shows the class name that should be used for the **objectType** input. **Note:** The class names can be different in UCMDB 7.0, 8.0, and 9.0.

### prop

A property for the **objectType** to add (name=value). For the **Add Object** operation, the key property for the type of CI to add must be entered if there are keys for that type of CI. For example, if you integrate with UCMDB 7.0 or 8.0, you must enter a CI with CIT **ip**, **prop ip\_address=19.35.55.34**. The **ip\_address** has to be unique in UCMDB. If you integrate with UCMDB 9.0, you must enter a CI with CIT **ip\_address**, **prop name=19.35.55.34**. The type of the property is determined automatically. You can add additional properties by creating extra prop inputs, each with a number appended to it, such as **prop1**. For example, if you integrate with UCMDB 7.0 or 8.0, enter **prop1 ip\_domain=battleground.ad** or **prop1 isvirtual=false**. If you integrate with UCMDB 9.0, enter **prop1 routing\_domain=battleground.ad** or **isvirtual=false**. You can find the property names in the CI Type Manager by clicking the **Attributes** tab for the CI item. The **Name** field shows the attribute name that should be used for prop input.

## Results

The operation returns the following results:

### returnResult

The ID of the created object.

## Add Relationship

The **Add Relationship** operation adds a Relation between two Configuration Items to the UCMDB based on the type of relation specified in the **relationType** input. It returns the ID of the created relationship.

## Inputs

All of the operation's inputs except the following are described in [Common Inputs in the Integration](#).

### relationType

The Configuration Item Type of the relation in the UCMDB. For example, if you integrate with UCMDB 7.0 or 8.0, **relationType contained** is the class name for the relation type with the display name **Contained**. If you integrate with UCMDB 9.0, **relationType containment** is the class name for the relation type with the display name **Containment**. You can find the relationship class names in the CI Type Manager by clicking the **Details** tab for the Relationships item. The **Name** field shows the class name that should be used for **relationType** input.

### fromId

The ID of the Configuration Item the relationship is from in the UCMDB.

### fromType

The Configuration Item Type the relationship is from in the UCMDB.

### told

The ID of the Configuration Item the relationship is to in the UCMDB.

### toType

The Configuration Item Type the relationship is to in the UCMDB.

### prop

A property for **relationType** to add (*name=value*). For example, if you integrate with UCMDB 7.0 or 8.0, you can specify **display\_label=My Contained** for **relationType contained**. If you integrate with UCMDB 9.0, you can specify **display\_label=My Containment** for **relationType containment**. The type of the property is determined automatically. You can add additional properties by creating extra **prop** inputs, each with a number appended to it, such as **prop1**. You can find the property names in the CI Type Manager by clicking the **Attributes** tab for the Relationships item. The **Name** field shows the attribute name that should be used for **prop** input.

## Results

The operation returns the following results:

### returnResult

The ID of the created relation.

## Delete Object

The **Delete Object** operation deletes a Configuration Item in the UCMDB based on the **objectId** and the **objectType** specified in the inputs.

## Inputs

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### objectId

The Configuration Item ID in the UCMDB.

### objectType

The Configuration Item Type of the UCMDB. For example, you can specify an **objectType** of **nt** which is the class name for the CIT with the display name **Windows**. You can find the class name in the CI Type Manager by clicking the **Details** tab for the CI item. The **Name** field shows the class name that should be used for the **objectType** input. **Note:** The class names can be different in UCMDB 7.0, 8.0, and 9.0.

## Delete Relation

The **Delete Relation** operation deletes a Relation between two Configuration Items in the UCMDB.

### Inputs

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

#### relationId

The Configuration Item ID for the Relation in the UCMDB.

#### relationType

The Configuration Item Type of the relation in the UCMDB. For example, if you integrate with UCMDB 7.0 or 8.0, **relationType contained** is the class name for the relation type with the display name **Contained**. If you integrate with UCMDB 9.0, **relationType containment** is the class name for the relation type with the display name **Containment**. You can find the relationship class names in the CI Type Manager by clicking the **Details** tab for the Relationships item. The **Name** field shows the class name that should be used for **relationType** input.

#### fromId

The ID of the Configuration Item the relationship is from in the UCMDB.

#### told

The ID of Configuration Item the relationship is to in the UCMDB.

## Get Filtered Object List by Type

The **Get Filtered Object List by Type** operation retrieves all of the Configuration Items that match the conditions specified in the UCMDB based on the **objectType**. It returns a list of the IDs of the Configuration Items.

### Inputs

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### objectType

The Configuration Item Type of the UCMDB. For example, you can specify an **objectType** of **nt** which is the class name for the CIT with the display name **Windows**. You can find the class name in the CI Type Manager by clicking the **Details** tab for the CI item. The **Name** field shows the class name that should be used for **objectType** input. **Note:** The class names can be different in UCMDB 7.0, 8.0 and 9.0.

### logicalOperator

Specify **AND** if all conditions must be matched or **OR** if only one or more of conditions must match.

### conditionAttributes

A comma-delimited list of the attribute names of the Configuration Item that the conditions test. For example, **ip\_ismanaged**, **isvirtual** are the attributes for the Configuration Item Type **IP** (**ip** as class name) in UCMDB 7.0 and 8.0, or **IpAddress** (**ip\_address** as class name) in UCMDB 9.0 with the display attribute names **IP Is Managed** and **Is Virtual**.

### conditionOperators

A comma-delimited list of the comparisons to do, corresponding to the input **conditionAttributes**. For example, **==,!=",like**.

### conditionValues

A comma-delimited list of the values to compare the property to, corresponding to the input **conditionOperators**. For example, **true,true**.

## Results

The operation returns the following results:

### CIs

The objects of that type that match the conditions.

## Get Neighboring Objects

The **Get Neighboring Objects** operation retrieves all the related Configuration Items based on **objectId** and the **objectType** in the UCMDB. It returns all the related Configuration Item IDs, relation types, and a topology map in XML format of the related CIs.

## Inputs

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### objectId

The Configuration Item ID in the UCMDB.

### objectType

The Configuration Item Type (CIT) of the UCMDB. For example, you can specify an **objectType** of **nt** which is the class name for the CIT with the display name **Windows**. You can find the class name in the CI Type Manager by clicking the **Details** tab for the CI item. The **Name** field shows the class name that should be used for **objectType** input. **Note:** The class names can be different in UCMDB 7.0, 8.0, and 9.0.

## Results

The operation returns the following results:

### objectIds

The IDs of the neighboring objects of that type.

### objectTypes

The types of neighboring objects.

### relationIds

The IDs of neighboring relations.

### relationTypes

The types of neighboring relations.

### topologyXML

The XML of the local topology (objects and types).

## Get Object Attributes by Id

The **Get Object Attributes by Id** operation retrieves the attributes specified in **attributeList** for the Configuration Item (CI) based on the **objectId** and the **objectType** specified in the inputs in the UCMDB. It returns the CI attributes in the format of `<name=value>` list.

## Inputs

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### objectId

The Configuration Item ID in the UCMDB.

### objectType

The Configuration Item Type of the UCMDB. For example, you can specify an **objectType** of **nt** which is the class name for the CIT with the display name **Windows**. You can find the class name in the CI Type Manager by clicking the **Details** tab for the CI item. The **Name** field shows the class name that should be used for **objectType** input. **Note:** The class names can be different in UCMDB 7.0, 8.0, and 9.0.

### attributeList

A comma-delimited list of attributes to retrieve in the UCMDB. For example, **attributeList** **host\_key,root\_lastaccesstime** contains the attributes for the Configuration Item Type **Windows** (or **nt** as class name) with display attribute names **Host Key** and **Last Access Time**. If you specify **none** for **attributeList**, no attributes are retrieved. If you leave **attributeList** blank, all of the attributes of that CIT are retrieved. You can find the attribute names in the CI Type Manager by clicking the **Attributes** tab for the CI item. The **Name** field shows the attribute name that should be used for **attributeList** input.

## Results

The operation returns the following results:

### Attributes

The attributes and their values.

## Get Object List By Host Name

The **Get Object List by Host Name** flow checks if the Configuration Item (CI) with **hostname** exists in UCMDB. If you integrate with UCMDB 7.0 or 8.0, it returns the IDs of the CIs in UCMDB with the attribute **host\_hostname,host\_key, host\_dnsname, or display\_label** that is equal to the input **hostname** and the CI type of the returned CIs. If you integrate with UCMDB 9.0, it returns the IDs of the CIs in UCMDB with **attribute name, primary\_dns\_name, host\_key, or display\_label** that is equal to the input **hostname** and the CI type of the returned CIs. The flow won't return all the CIs that have the same hostname but belong to different CI types. This flow is intended to be used as a subflow for the flow **Validate IP** in UCMDB.

## Inputs

All of the flow's inputs except the following are described in [Common inputs in the integration](#).

### hostname

The host name to check in UCMDB. For example, myserver.rose.hp.com or myserver.

### cmdbHostTypeList

The list needed to include UCMDB CI types that belong to **host** CI types if you integrate with UCMDB 7.0 and 8.0, or **node** CI types if integrate with UCMDB 9.0. Start with the host or node type that is more specific. For example, nt,unix. If not specified, the following default list will be used: nt,unix,vmware\_esx\_server, cluster\_resource\_group, clustered-service,terminalserver,vax,mainframe,lpar,host\_node,node,host. node and cluster\_resource\_group are the class names for UCMDB 9.0. We recommended that you not leave this input empty.

## Results

The operation returns the following results:



`returnHostCIResult`

The IDs of the CIs in UCMDB with the attribute **host\_key** or **host\_hostname**, or **display\_label** that is equal to the input **hostname**.

`returnHostCITypeResult`

The CI type of the **returnHostCIResult**.

## Get Object List by Type

The **Get Object List by Type** operation retrieves all the Configuration Items in the UCMDB based on the **objectType** specified in the input. It returns a list of CI IDs for the given Configuration Item Type.

### Inputs

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

`objectType`

The Configuration Item Type of the UCMDB. For example, you can specify an **objectType** of **nt** which is the class name for the CIT with the display name **Windows**. You can find the class name in the CI Type Manager by clicking the **Details** tab for the CI item. The **Name** field shows the class name that should be used for **objectType** input. **Note:** The class names can be different in UCMDB 7.0, 8.0, and 9.0.

### Results

The operation returns the following results:

**CIs**

The objects of that type.

## Get Query Name of View

The **Get Query Name of View** operation retrieves the query name in the UCMDB for the given **viewName**.

### Inputs

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

`viewName`

The view name in the UCMDB. For example, **NetworkTopology** is one of the default views in the UCMDB.

### Results

The operation returns the following results:

`returnResult`

The name of the query.

## Get Topology Map by Query Name

The **Get Topology Map by Query Name** operation retrieves the Topology Map in XML format for the given query name in the UCMDB.

### Inputs

All of the operation's inputs except the following are described in *Common inputs in the integration*.

`queryName`

The query name in the UCMDB. For example, **NetworkTopology** is one of the default queries in the UCMDB.

### Results

The operation returns the following results:

`topologyXML`

The XML of the local topology (objects and types).

## Get Server Status

Retrieve current status of the UCMDB server, to check if it is available. Also retrieve current status of the System Capacity Monitors and the Performance Monitors. Currently this operation is only available and supported for UCMDB 9.x.

### Inputs

All of the operation's inputs are described in *Common inputs in the integration*.

### Responses

The operation returns the following results:

`success`

The UCMDB server status is successfully retrieved.

`failure`

The operation failed with thrown exception.

### Results

The operation returns the following results:

returnResult

This is the primary output.

serverStatus

The status info of the server monitors, including the Availability Monitors, the System Capacity Monitors, and the Performance Monitors.

## Get Topology Map by Query Name with Parameter

The **Get Topology Map by Query Name with Parameter** operation retrieves a topology for a query you set up with a parameterized value. For example, if want to restrict the query to list only **display\_label=10.22.0.0** for the network node, create a query in UCMDB with the Network Configuration Item Type that has an attribute of **Display Label** with a parameterized value type.

### Inputs

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

queryName

The query to run. For example, you can set up a query called **NetworkTopologyWithParameters** which allows filtering parameters in the UCMDB.

parameterName

The parameter name to filter on the attributes of in the UCMDB.

stringProp

The parameter to pass, which must be of the type **string**. For example, **display\_label=10.22.0.0**. If you do not specify a value, the query returns everything without restriction. Additional properties can be added by adding extra **stringProp** inputs each with a sequential number appended to it, such as **stringProp1**. The same rule applies to the props of different data types.

floatProp

A parameter to pass of type **float**.

booleanProp

A parameter to pass of type **boolean**.

integer\_listProp

A parameter to pass that is a comma-delimited list of integers.

longProp

A parameter to pass of type **long**.

xmlProp

A parameter to pass of type **xml**.

`string_listProp`

A parameter to pass that is a comma-delimited list of strings.

`integerProp`

A parameter to pass of type **int**.

`dateProp`

A parameter to pass of type **date**.

`bytesProp`

A parameter to pass, as a hex number.

`doubleProp`

A parameter to pass of type **double**.

`parameterName`

The name of the parameter to filter the attributes on. This is for internal use only. It doesn't affect the query.

## Results

The operation returns the following results:

`topologyXML`

The XML of the local topology (objects and types).

## Get Topology Map by View Name

The **Get Topology Map by View Name** operation retrieves the Topology Map in XML format for the specified view name from the UCMDB.

## Inputs

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

`viewName`

The view name in the UCMDB. For example, **NetworkTopology** is one of the default views in the UCMDB.

## Results

The operation returns the following results:

`topologyXML`

The XML of the local topology (objects and types).

## Is Object Related To

The **Is Object Related To** flow checks to see if one Configuration Item (CI) ID is related to another CI ID in UDMCB. It returns **Success** if they are related. This flow is intended to be used as a subflow for the flow **Validate IP** in UCMDB.

### Inputs

All of the flow's inputs except the following are described in [Common inputs in the integration](#).

#### `objectIdToRelate`

The CI ID in UCMDB to check if it is related to the input **objectIdRelateTo**.

#### `objectIdRelateTo`

The CI ID in UCMDB to check if the input **objectIdToRelate** has a relationship with this id. This CI ID has to be from CI in the relationship in order to get neighboring objects.

#### `objectTypeOfObjectIdRelateTo`

The CI type in UCMDB for the input **objectIdRelateTo**. For example, you can specify an **objectType** of **nt or unix**, which is the class name for the CIT with the display name **Windows or Unix**.

## Modify Object Attribute(s)

The **Modify Object Attribute(s)** operation modifies the attributes of the Configuration Item in the UCMDB based on the **objectId** and the **objectType** specified in the inputs.

### Inputs

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

#### `objectId`

The Configuration Item ID in the UCMDB.

#### `objectType`

The Configuration Item Type of the UCMDB. For example, you can specify an **objectType** of **nt** which is the class name for the Configuration Item Type with the display name **Windows**. You can find the class name in the CI Type Manager by clicking the **Details** tab for the CI item. The **Name** field shows the class name that should be used for **objectType** input. **Note:** The class names can be different in UCMDB 7.0, 8.0, and 9.0.

## prop

A property for the **objectType** to modify (*name=value*). The type of the property is determined automatically. You can add additional properties by creating extra **prop** inputs, each with a number appended to it, such as **prop1**. For example, if you integrate with UCMDB 7.0 or 8.0, **prop ip\_domain=battleground.ad** or **prop1 isvirtual=false**. If you integrate with UCMD 9.0 **prop routing\_domain=battleground.ad** or **prop1 isvirtual=false**. You can find the attribute names in the CI Type Manager by clicking the **Attributes** tab for the CI item. The **Name** field shows the attribute name that should be used for **prop** input.

## Validate IP in UCMDB

The **Validate IP in UCMDB** flow looks up the IP for **hostname** from DNS server. If it finds the IP, it checks if the Configuration Item (CI) with that IP exists in UCMDB. If the IP exists, it checks to see if the CI with **hostname** exists in UCMDB. If **hostname** exists it checks to see if the **host** CI is related to the **ip** CI. The operation returns the found host CI type, host CI ID, and ip CI ID.

## Inputs

All of the flow's inputs except the following are described in [Common inputs in the integration](#).

### hostname

The fully-qualified domain name. For example, myserver.rose.hp.com.

### dnsServer

This is an optional input. Specifies the DNS server name or ip to look up the IP for the input **hostname**.

### cmdbHostTypeList

A list that includes UCMDB CI types that belong to **host** CI types if you integrate with UCMDB 7.0 and 8.0, or **node** CI types if you integrate with UCMDB 9.0. Start with the host or node type that is more specific. For example, **nt,unix**. If not specified, the following default list is used: nt,unix,vmware\_esx\_server, cluster\_resource\_group, clusteredservice,terminalserver,vax,mainframe,lpar,host\_node,node,host. node and cluster\_resource\_group are the class names for UCMDB 9.0. We recommend that you not leave this input empty.

## Results

The operation returns the following results:

### returnIpCIResult

The ID of the CI in UCMDB with an **ip\_address** that is equal to the IP of the input hostname.

### returnHostCITypeResult

The CI type of the host to which the **returnIpCIResult** is related.

`returnHostCIResult`

The ID of the host CI in UCMDB to which the **returnIpCIResult** is related.

---

## 4 Troubleshooting

This section includes the following topics:

- [Troubleshooting Overview](#)
- [General Troubleshooting Procedures and Tools](#)
- [Error Messages](#)



# Troubleshooting Overview

This section provides troubleshooting procedures and tools you can use to solve problems you may encounter while using this integration. It also includes a list of the error messages you may receive while using the integration and offers descriptions and possible fixes for the errors.

## General Troubleshooting Procedures and Tools

This section describes the troubleshooting procedures and tools you can use to fix problems you may experience while using this integration.

### If a flow fails when run in OO Central

- 1 In Operations Orchestration Central, click the **Reports** tab and select **Error** and **Failed** in the **Results** list, or select nothing.

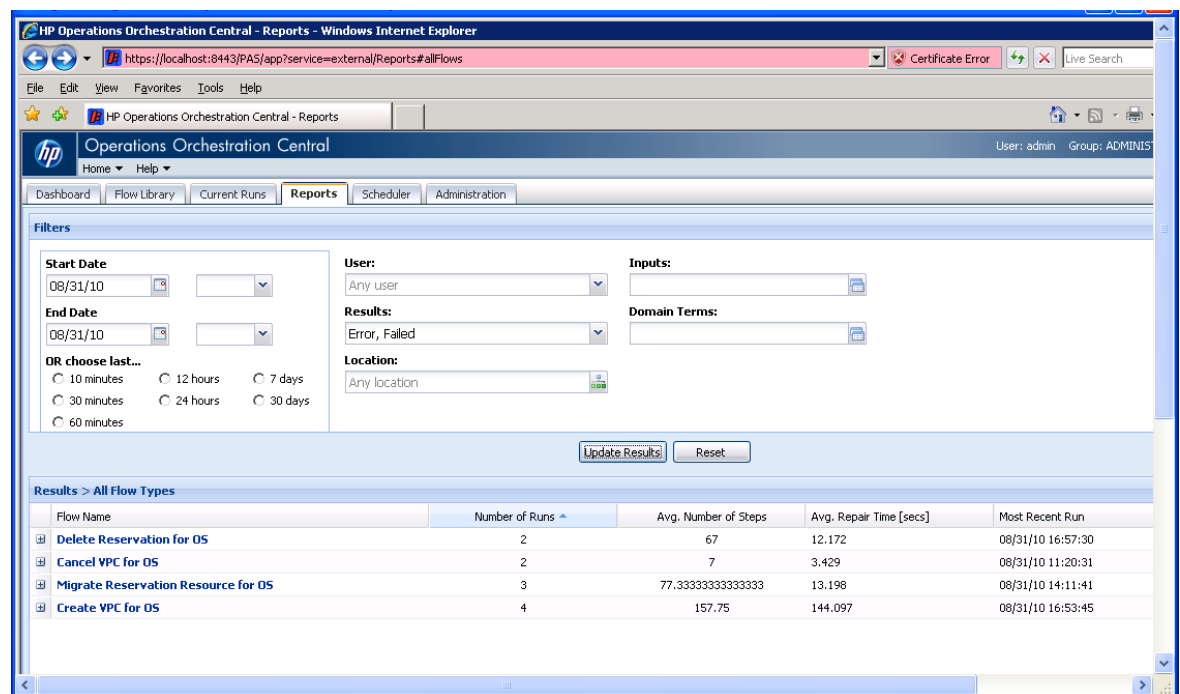


Figure 2 - OO Central Reports tab

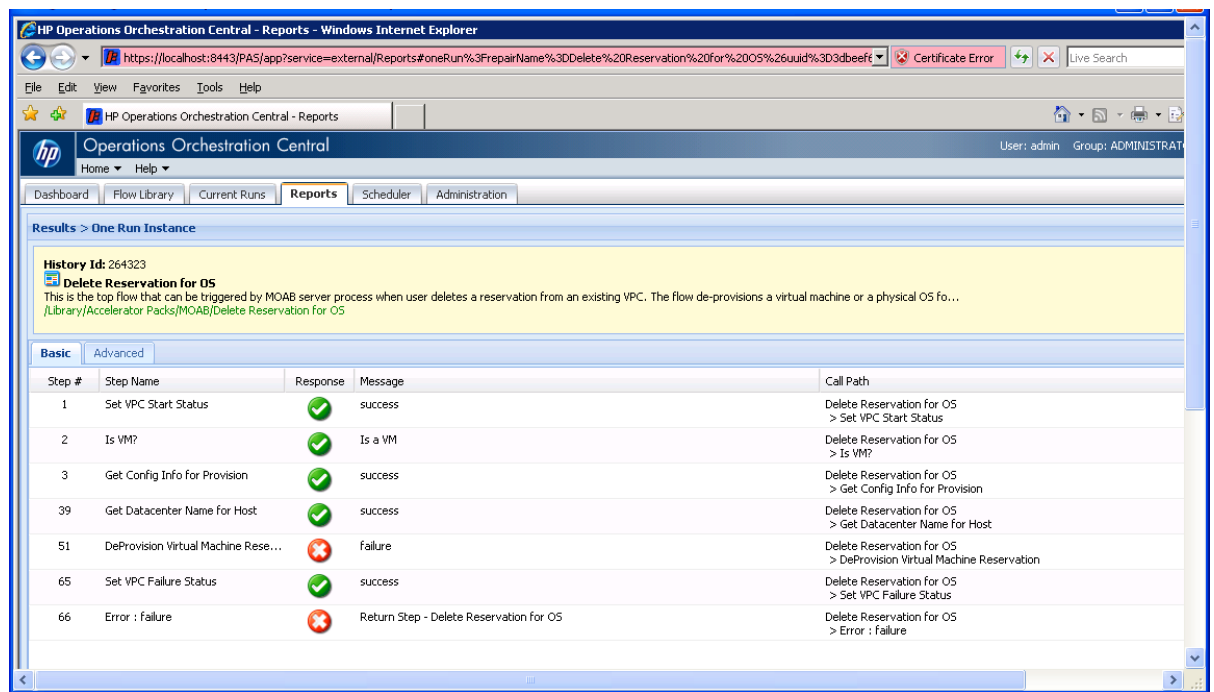
- 2 Find the failed flow you want to examine and click it.

The screen changes to show you the history of the runs of the flow.

- 3 Click the **History ID** of the job you want to view.

The list of History IDs is based on time.

For example, in the above figure, you could click **Delete Reservation for OS** and then click the **History ID** of the job that failed.



**Figure 3 - OO Central History ID**

- 4 Locate the failure step.  
For example, in the above figure note that step number 51 failed.
- 5 To research the failure, click the **Advanced** tab.

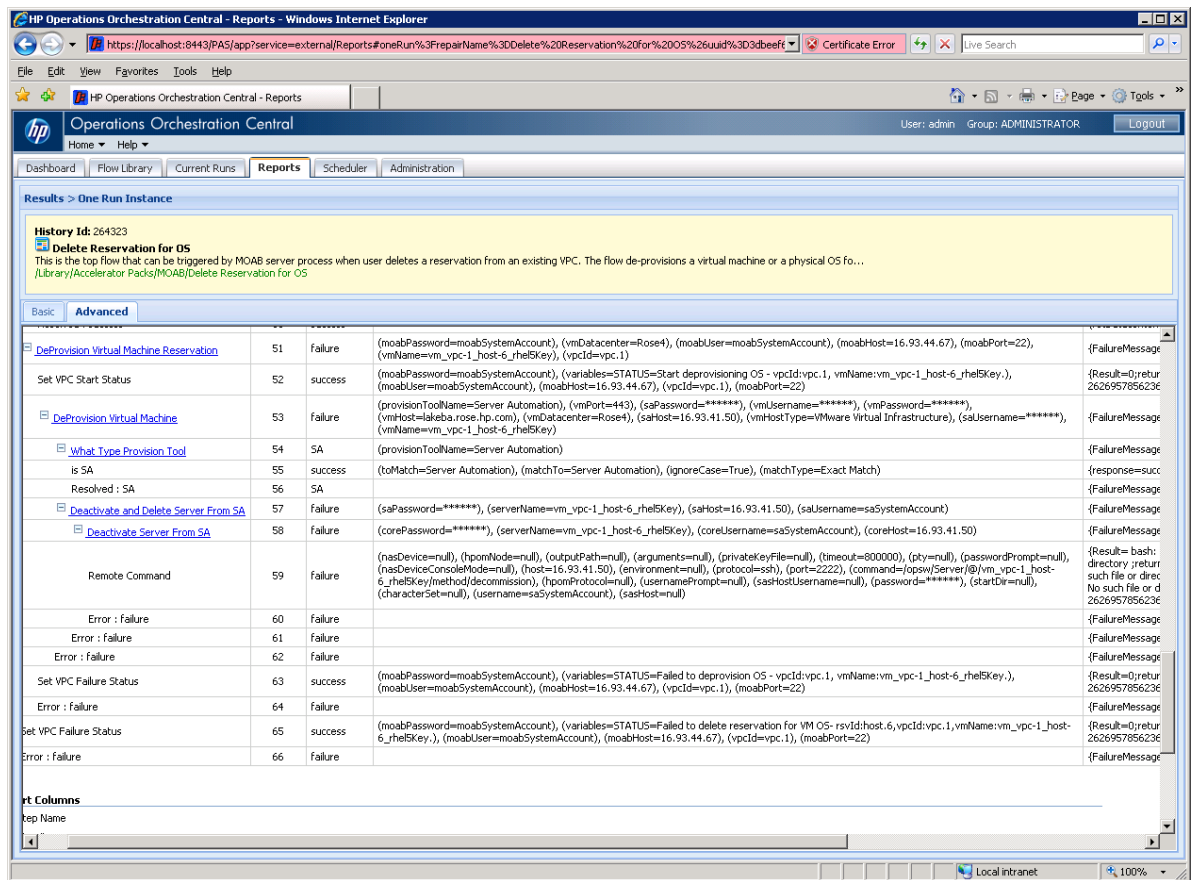


Figure 4 - OO Central History ID Advanced tab

- 6 In the Report Columns area, select the Step Name, Step #, Response, Bound Inputs, and Result check boxes.

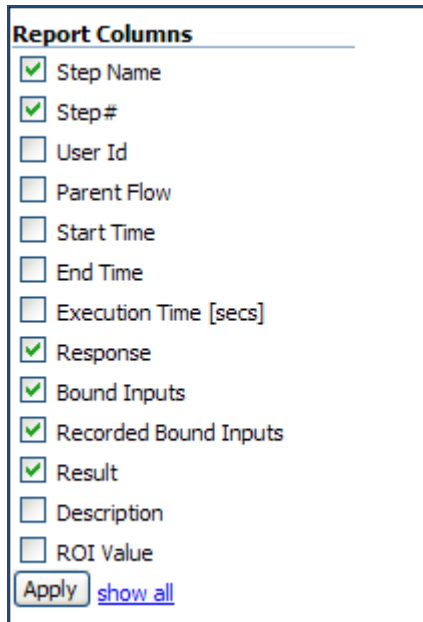


Figure 5 - Report Columns area

- 7 Find step 51 and then find the actual step that failed.
- 8 Examine the error message from the result.

If you run the flow in OO Studio using the **Debug** feature, you can use the **Step Result Inspector** to check the errors and exceptions in detail.

Similar information can also be found in the OO Central log `Central_wrapper.log` located in the OO home folder in `/Central/logs/`. For information about the RAS service, check the log file `wrapper.log` located in the OO home folder in `/RAS/Java/Default/webapp/logs`.

## Error Messages

This section lists the error messages you may receive while using this integration. Each error message includes possible causes and fixes for the error.

`org.apache.axis2.AxisFault Transport error: 404 Error: Path+Not+Found`

This message means that the logon information for UCMDB 7.0 may be incorrect. Check the **cmdbHost** and **cmdbPort** inputs to make sure that they are correct.

`com.hp.ucmdb.api.CommunicationException: Got response code 404 for URL http://10.51.0.198:8080/ucmdb-api/connect?VERSION=8.0`

This means that the logon information for UCMDB 8.0 may be incorrect. Check the **cmdbHost** and **cmdbPort** inputs to make sure that they are correct.

`org.apache.axis2.AxisFault Transport error: 401 Error: Unauthorized`

This means that the logon information for UCMDB 7.0 may be incorrect. Check the **username** and **password** inputs to make sure that they are correct.

`Invalid credentials`

This means that the logon information for UCMDB 8.0 may be incorrect. Check the **username** and **password** to make sure that they are correct.

`No enum const class`

`com.hp.oo.content.connectors.hp.ucmdb.AbstractUCMDBConnector$versions.v10`

The **cmdbVersion** string is incorrect. The valid strings are **7**, **8**, and **9**.

`class [Windows] not found in CMDB Class Model; nested exception is: java.lang.Exception:`

The **objectType** string is not defined in UCMDB 7.0. Make sure the type is defined as the **class\_name** in the **dbo.ccm\_classes** table of <CMDB Schema>. In the UCMDB GUI, the Configuration Item Type (CIT) shows the type's display name. For example, for the CIT display name **Windows**, the actual class name is **nt**.

`com.hp.ucmdb.api.classmodel.ClassDoesNotExistException: Class "Windows" is not defined in the uCMDB class model`

The **objectType** string is not defined in UCMDB 8.0. Make sure the type is defined as the **class\_name** in the **dbo.ccm\_classes** table of <CMDB Schema>. In the UCMDB GUI, the Configuration Item Type (CIT) shows the type's display name. For example, for the CIT display name **Windows**, the actual class name is **nt**.

The received string `ec928b8e1398f73b5011067b404088` representation is inconsistent; nested exception is: `java.lang.Exception`:

The **objectId** doesn't exist in UCMDB 7.0. Make sure that the ID is valid.

ID `5bf303466b8a982601dd3a9e88ad75` being restored has incorrect length (not 32 characters)

The **objectId** entered does not exist in UCMDB 8.0. Make sure that the ID is valid.

Object with ID: `5bf303466b8a982601dd3a9e88ad7557` was not found

The **objectId** entered does not exist in UCMDB 8.0. Make sure that the ID is valid.

Element Number: 1 doesn't exist in the result.; nested exception is: `java.lang.Exception`:

When you executed the operation **Get Neighboring Objects**, the **objectId** was not correct in UCMDB 7.0. Make sure that the ID is valid.

CMDB Operation Internal Error:... class name: Windows doesn't exist in class model

When you executed the operation **Get Neighboring Objects**, the **objectId** was not correct in UCMDB 7.0. Make sure that the ID is valid.

No Error Messages, but Got Empty Topology Map of Neighbors Cls.

When running the OO operation **Get Neighboring Objects**, the **objectId** was not correct in UCMDB 8.0. Make sure that the ID is valid.

Unable to determine the type of attribute: `ip_isManaged` because it is not a member of the class: `ip`

Make sure that you enter the **conditionAttributes** with valid attribute names. The attribute names can be found in the **attribute\_name** in the **dbo.ccm\_attributes** table of <CMDB Schema>.

Unknown Comparison type: `pp`

Make sure that you enter the **conditionOperators** with valid condition operators. Valid condition operators include `==`, `!=`, `<>`, `>`, and `<`.

`java.lang.Exception`: The specified condition type is not supported for: `BooleanCondition`, Supported Types are: `EqualNotEqual`

Make sure that the **conditionOperators** match with **conditionAttributes** type. For example, **conditionAttributes** `ip_ismanaged`, should have a **conditionOperator** of `==` or `!=`.

The lists of conditional types, names, operators and values are not of equal length

Make sure that the **conditionOperators**, **conditionAttributes**, and **conditionValues** are all the same length.

Unable to locate view `badViewName`

Make sure to enter a valid view name of the UCMDB in **viewName**.

Pattern [`badQueryName`] doesn't exist

Make sure to enter a valid query name of the UCMDB in **queryName**.

No Error Messages, but Got Empty Topology Map

If you get an empty topology map based on the **viewName** or **queryName**, the view or query might be empty in UCMDB.

Error occurred while trying to execute operation Data In - Add Data Strict : operation Data In - Add Data Strict....Couldn't append existing object:

When running the OO operation **Add Object** to create a Configuration Item, the Configuration Item is already in UCMDB 8.0. Make sure that the CI is not an existing CI.

the ID property [attribute name='ip\_address'] doesn't exist !!!; nested exception is:  
java.lang.Exception:

When running the OO operation **Add Object** to create a Configuration Item (CI) in UCMDB 7.0, the **prop** field should be the ID attribute. For example, if you create a type **ip** CI, the ID attribute is **ip\_address**; if you create a **host** CI, the ID attribute is **host\_key**.

Error occurred while trying to execute operation Data In - Add Data Strict : operation Data In - Add Data Strict.... Can't achieve ID properties: not all the requested ID properties exist

When running the OO operation **Add Object** to create a Configuration Item (CI) in UCMDB 8.0, the **prop** field should be the ID attribute. For example, if you create a type **ip** CI, the ID attribute is **ip\_address**; if you create a **host** CI, the ID attribute is **host\_key**.

Error occurred while trying to execute operation Data In - Add Or Ignore Data : operation Data In - Add Or Ignore Data ... Caused by: java.lang.NullPointerException

When running the OO operation **Add Relationship** to create a Relation in UCMDB 8.0, the relationship type might be not correct.

---

## 5 Security

This section includes the following topic:

- [About UCMDB Security](#)

## About UCMDB Security

UCMDB 7.0 servers are accessed via SOAP over HTTP (or HTTPS, if enabled on the host). UCMDB 8.0 servers are accessed via JAVA Client API. The UCMDB administrator provides logon credentials for connecting with the SOAP or API. The SOAP client or API client needs the username and password of an integration user defined in the UCMDB. These users do not represent actual users of the UCMDB, but rather applications that connect to the UCMDB.



---

## 6 OO Tools

This section includes the following topic:

- [OO Tools You Can Use with the UCMDB-OO Integration](#)

# OO Tools You Can Use with the UCMDB-OO Integration

Following are OO tools that you can use:

- **RSFlowInvoke.exe and JRSFlowInvoke.jar**

RSFlowInvoke (RSFlowInvoke.exe or the Java version, JRSFlowInvoke.jar) is a command-line utility that allows you to start a flow without using Central (although the Central service must be running). RSFlowInvoke is useful when you want to start a flow from an external system, such as a monitoring application that can use a command line to start a flow.

- **Web Services Wizard (wswizard.exe)**

When you run the Web Services Wizard, you provide it with the WSDL for a given Web service. The WSDL string you provide as a pointer can be a file's location and name or a URL. The Web Services Wizard displays a list of the methods in the API of the Web service that you specify. When you run the wizard, pick the methods you want to use, and with one click for each method you have selected, the wizard creates an HP OO operation that can execute the method. This allows you to use the Web Services Wizard to create operations from your monitoring tool's API.

These tools are available in the Operations Orchestration home folder in /Studio/tools/.