CISCO IMC Remote Action Service 1.1.1

For Hewlett Packard Operations Orchestration 9.00

User Guide

April 04, 2014

Table of Contents

Table of Figures

1 Overview

The CISCO IMC Remote Action Service (RAS) is an add-on for Hewlett-Packard Operations Orchestration 9.0 (HPOO) that enables the user to automate tasks in the Cisco IMC which includes both C-Series and E-Series environment.

User can use this RAS to create Flows that interact with and transfer information to Cisco IMC to automate various configuration/maintenance tasks. A flow is a collection of HPOO operation that are linked together to perform a composite task.

2 System Requirements

Cisco IMC Remote Action Service is supported on

1. Hewlett-Packard Operations Orchestration 9.00 on Windows

For the System Requirements of HPOO refer to the HPOO's System Requirements Guide (See Reference section).

3 Supported Servers and Cisco IMC Versions

All Cisco standalone C-series and E-series are supported.

For C-series - Cisco IMC version 1.5 and above is supported.
For E-series - Cisco IMC version 2.2 and above is supported.

4 Configuring IMC RAS in HPOO

The user must have Cisco IMC Operations/Flows in the HPOO Central repository in order to be able to run them. The IMC Operations/Flows are provided in a repository zip file. This repository zip is to be unzipped and then imported into the required user repository.
For the end user to run these flows, the repository should be imported to the HPOO Central repository.
In case this repository is to be used by the developers, it should be imported to the HPOO Studio repository. Once the development is done, developer can publish this HPOO Studio repository to the HPOO Central repository for end user's use.

Note: The following tasks are to be performed by a user with administrative privileges.
This section describes the following tasks:

- Creating a new HPOO Studio repository
- Importing the repository containing IMC Operations/Flows into Studio/Central repository in HPOO installation
- Copying IMC Operations files in HPOO
- Importing Operations
- Publishing/Committing data from HPOO Studio repository to Central repository

4.1 Creating HPOO Studio repository

HPOO Studio repository can be used for development work. User can import new Operations or modify existing Operations in Studio repository. Flow development is also done in Studio repository.

To create a new HPOO Studio repository, follow steps mentioned below:

1. Launch HPOO and in the main menu click **"Repository"** -> **"Add Repository"**.

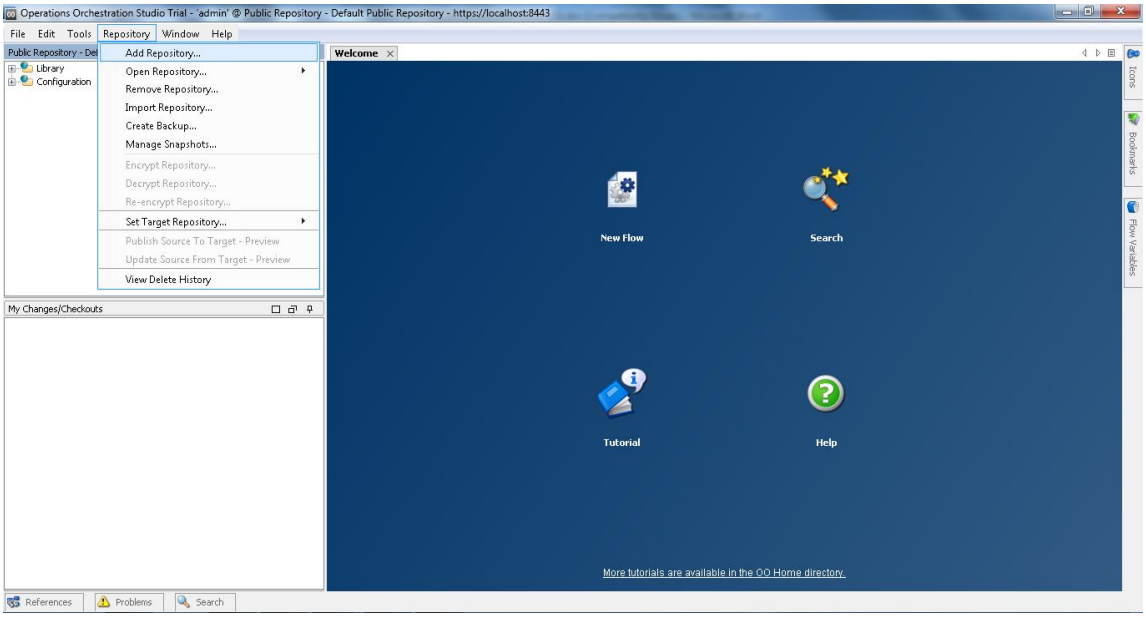

Figure 1: Add Repository

2. Enter the name and path of the repository to be created in **"Add Repository"** dialog-box and click the **"OK"** button.

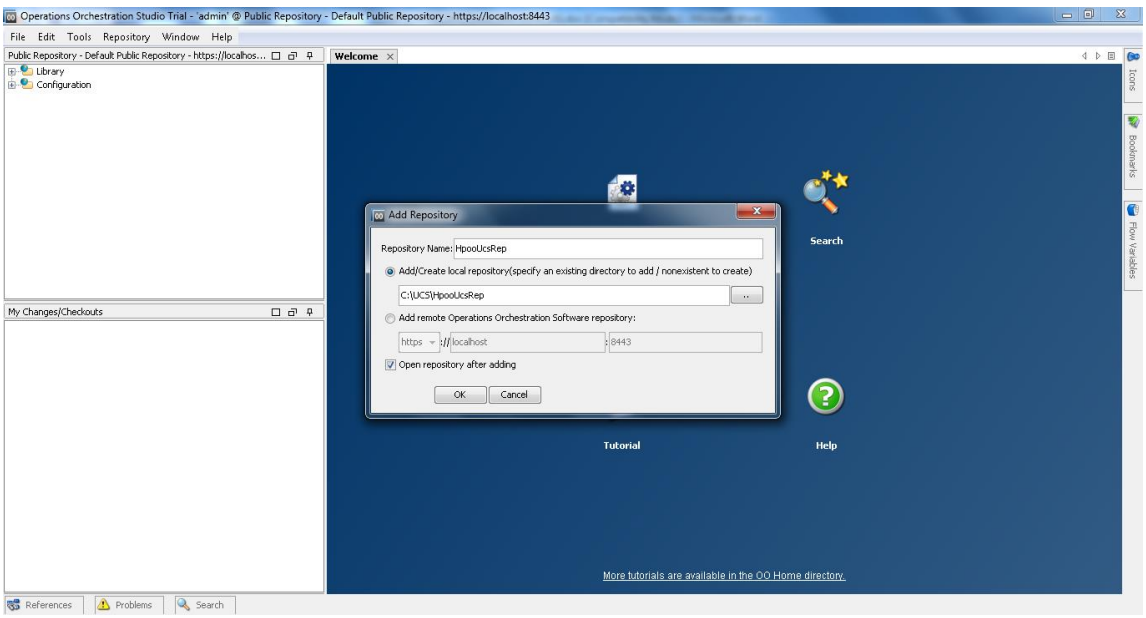

Figure 2: Add Repository Name and Path

Note: Please wait for few minutes as Repository takes some time to get created and loaded.

4.2 Importing IMC Operations and Flows from Repository

For importing the IMC Operations/Flows, a repository folder **"CiscoImcRepository"** is available to the end user in **"Cisco IMC Remote Action Service ver1.1.1.zip"**.
For importing UCS Operations/Flows from this repository, perform the following steps:-

1. Unzip **"Cisco IMC Remote Action Service ver1.1.1.zip"** .This would create a folder "Cisco IMC Remote Action Service ver1.1.1".
2. In the main menu, click **"Repository"** -> **"Import Repository"**.

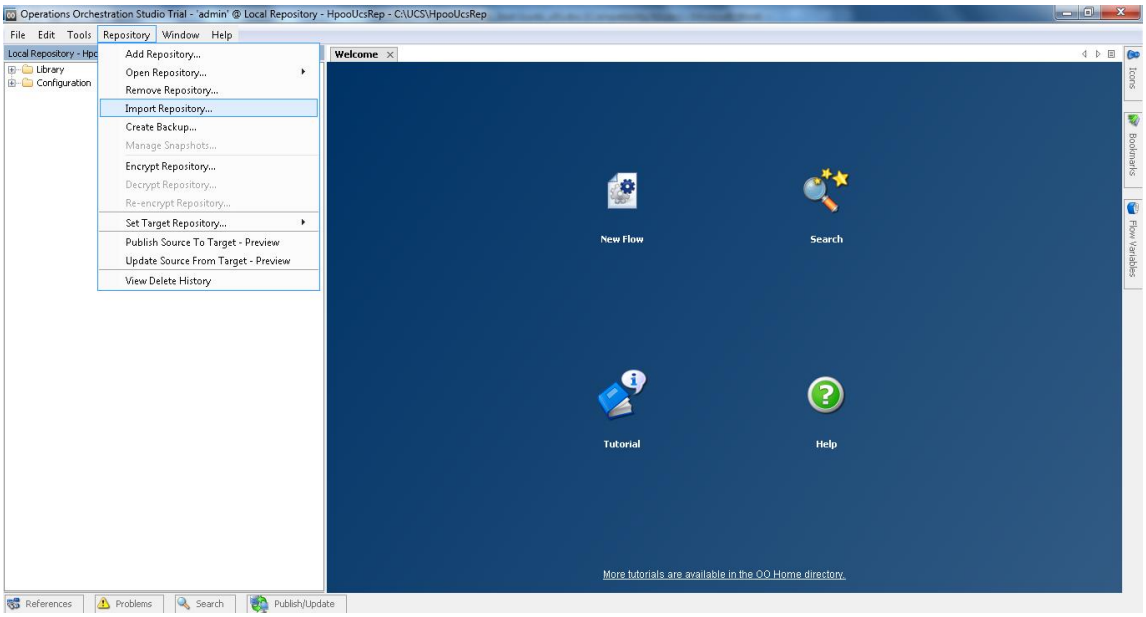

Figure 3: Import Repository

3. Enter the path of the folder unzipped in step 1; select CiscoImcRepository folder containing IMC Operations & Flows, and click the **"Open"** button.

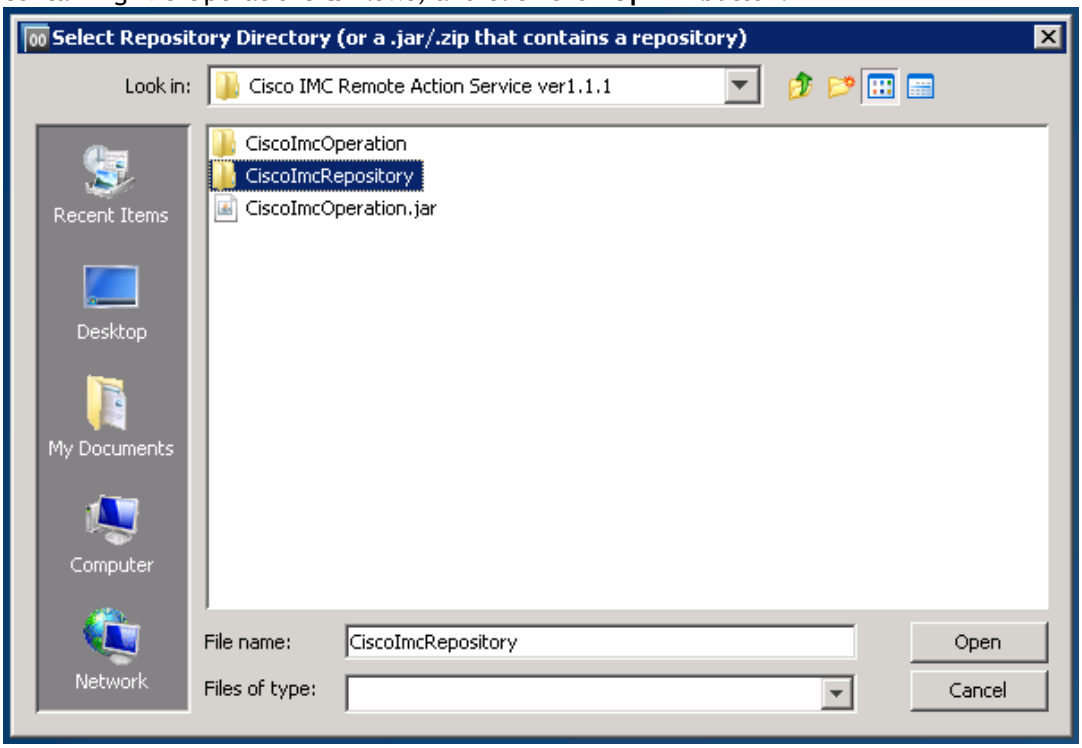

Figure 4: Select Import Repository Path

4. After the import repository path is selected, a window is displayed with the information showing the difference between the existing and the imported repository. User can verify the difference and then apply the changes to the existing repository.

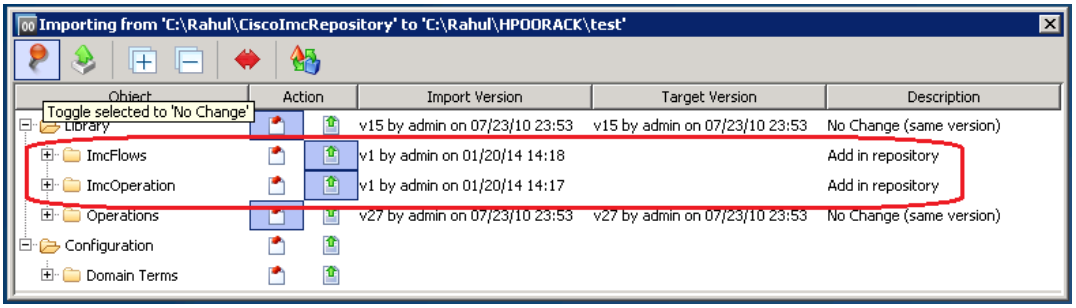

Figure 5: Import Repository Verify and Apply Changes

5. Once done, the IMC Operations/Flows and Configuration data from the imported repository is displayed in the current Studio directory as shown in Figure 6. Folders **"ImcOperation"**, **"ImcFlows"** in Library and selection list entries in Configuration would be newly selected.

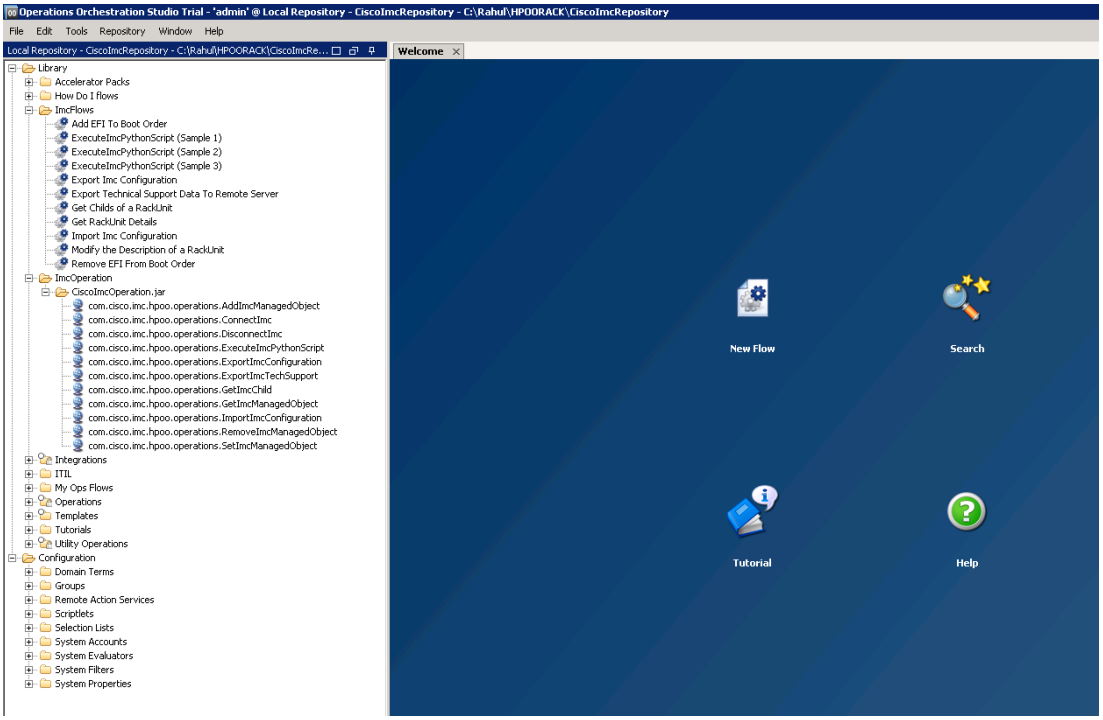

Figure 6: Imported Operations and Flows

Note: Steps for importing repository are common for both Studio and Central repository.

4.3 Copying IMC Operations Files in HPOO

For the imported IMC Operations/Flows to work successfully, user needs to copy the following jar files and folders to the appropriate locations in the HPOO installation directory structure.

Note: %ICONCLUDE_HOME% is the environment variable for HPOO home directory. Its default value is *"C:\Program Files\Hewlett-Packard\Operations Orchestration"*, but may vary depending on the installation of HPOO. Refer environment variables of host operating system for verification.

Operations Jar - This jar contains class files for all the IMC Operations which are imported in the HPOO repository. This *CiscoImcOperation.jar* needs to be copied at the following location: *"%ICONCLUDE_HOME%\RAS\Java\Default\repository"*.

Library Files - These files contains all classes related to common library functions needed to support the IMC Operations/Flows. The directory CiscoImcOperation (containing *CiscoImcSdk-0.9.0.jar*, jython.jar and Core.py files) needs to be copied at the following location: *"%ICONCLUDE_HOME%\RAS\Java\Default\repository\lib"*

Note: Currently only default RAS server is supported for IMC operations RAS, hence the files can be copied (installed) in the above mentioned paths only.

After copying jars, user needs to restart the following HPOO services via windows **Services**.

a. *RSCentral*

b. *RSJRAS*

4.4 Importing Operations

This section describes how to import new or modified operations in a repository where the IMC Operations/Flows already exist. The following steps are to be performed for importing these operations:

1. Complete the steps shown in section 3.3.
2. On the HPOO UI, select "**ImcOperation**" in the Repository pane.

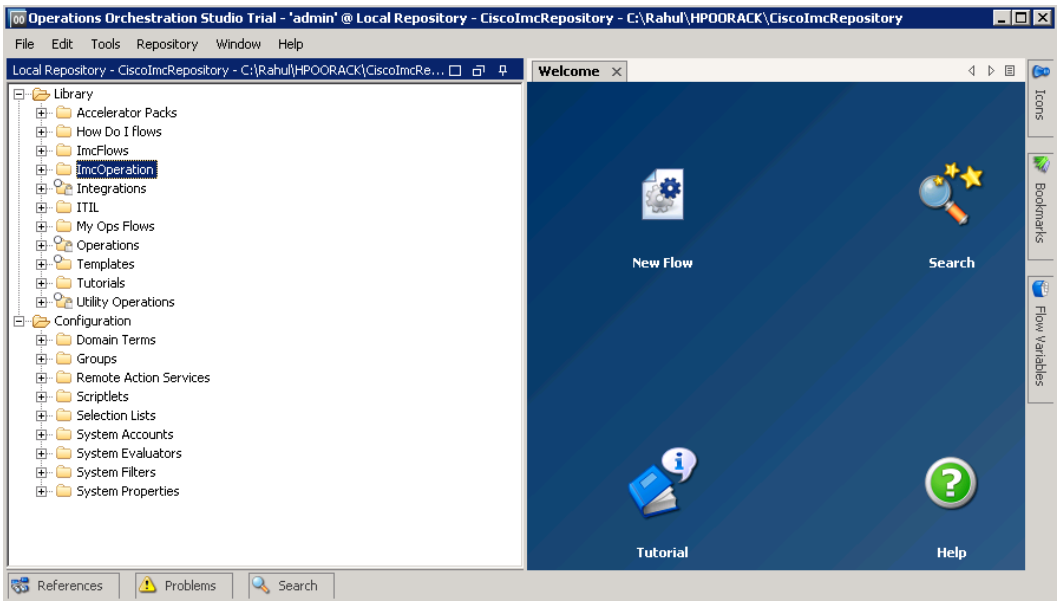

Figure 7: Import Added/Modified Operations

3. In the main menu, click **"File" -> "Create Operations from RAS…"**

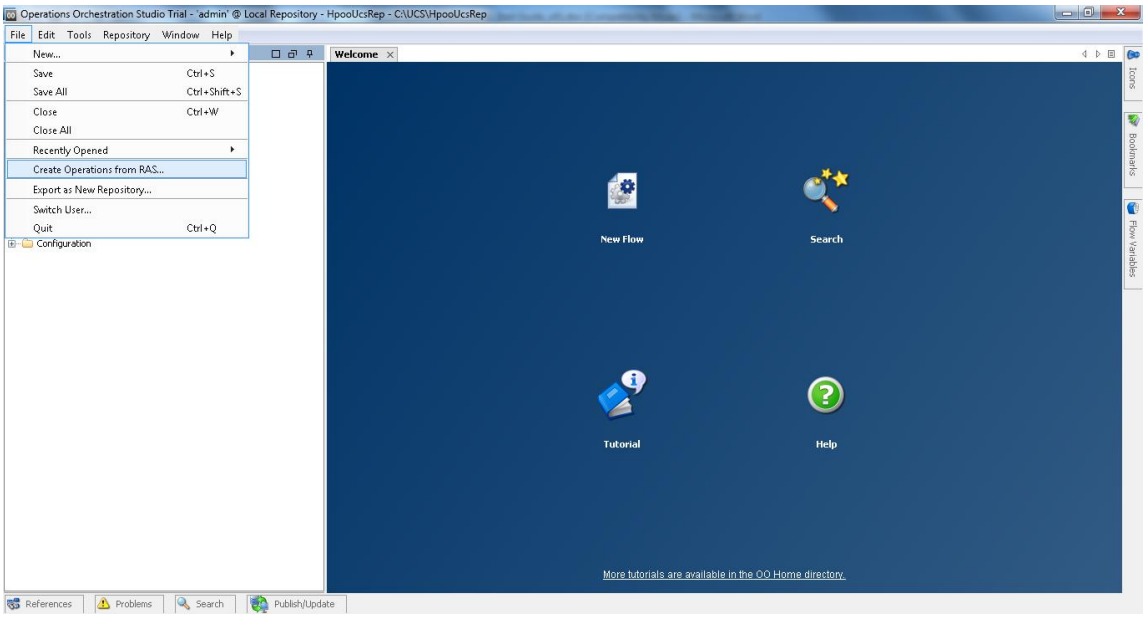

Figure 8: Create Operation from RAS

4. In the populated **"RAS Import"** dialog box, select the **"RAS_Operator_Path"** from the drop down menu option and click **"OK"**.

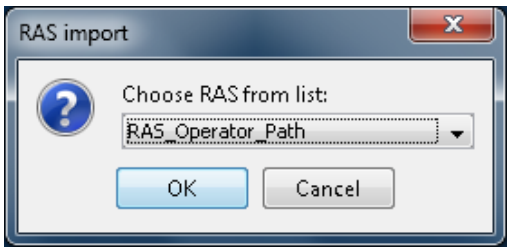

Figure 9: Create Operation from RAS Import

5. Select "*CiscoImcOperation.jar*" in the **"Create operations from RAS"** window and click **"OK"**.

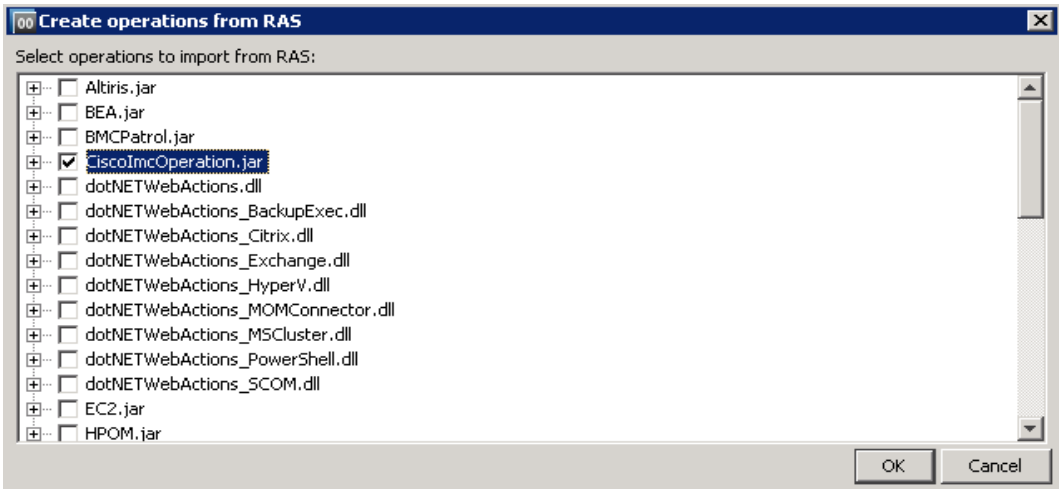

Figure 10: Create Operation from RAS IMC Operations

6. Select one or more options in the **"Operations Orchestration Studio"** window based on what operations are to be added/ modified, and click **"OK"**.

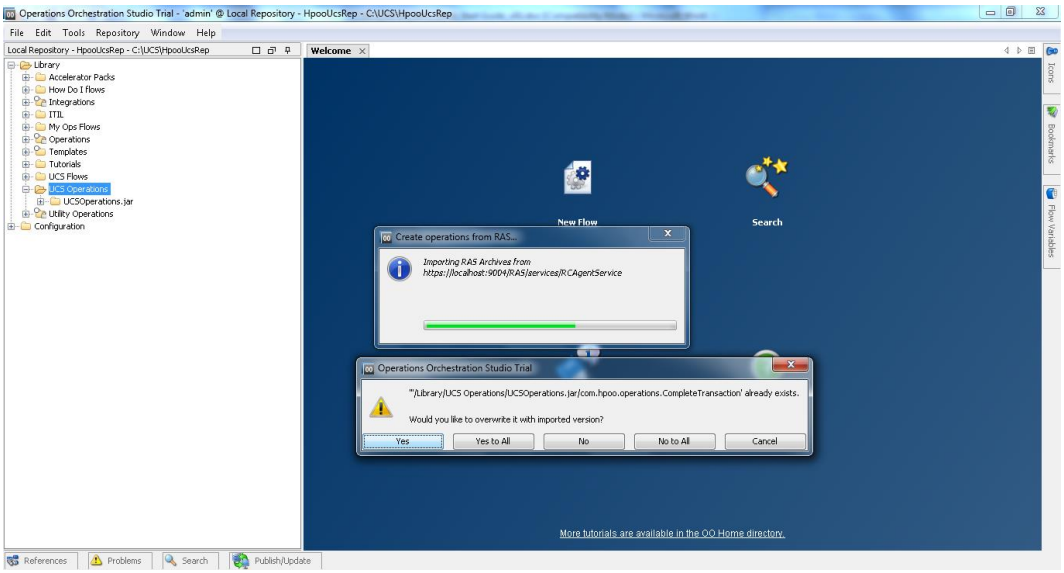

Figure 11: Create Operation from RAS Apply Changes

7. The added/modified operations should now be present in the **"ImcOperations"** folder.
   Open all operations one by one, and click on **Outputs tab** and in the **"Extract Primary Output From Field"** drop down, select **operationOutput**.

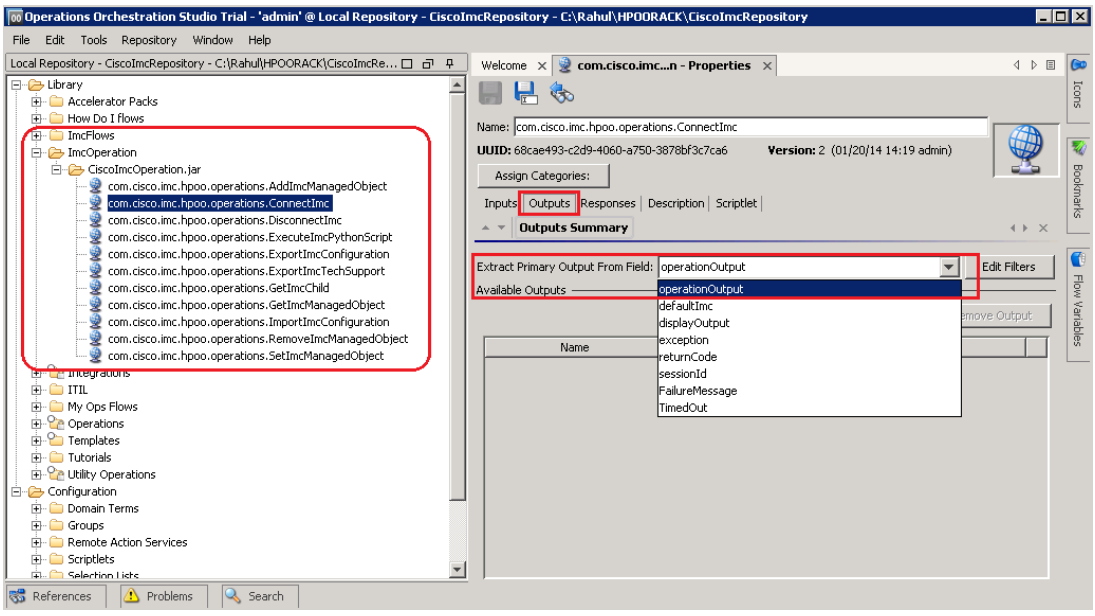

Figure 12: Create Operation from RAS Apply Changes

8. Repeat Step 7 for all operations and save the changes.

9. Open all operations one by one, and click on **Input tab**. Open **operationInput** field from input summary, and in **Otherwise** drop down select **"Use Previous Step Result"**.

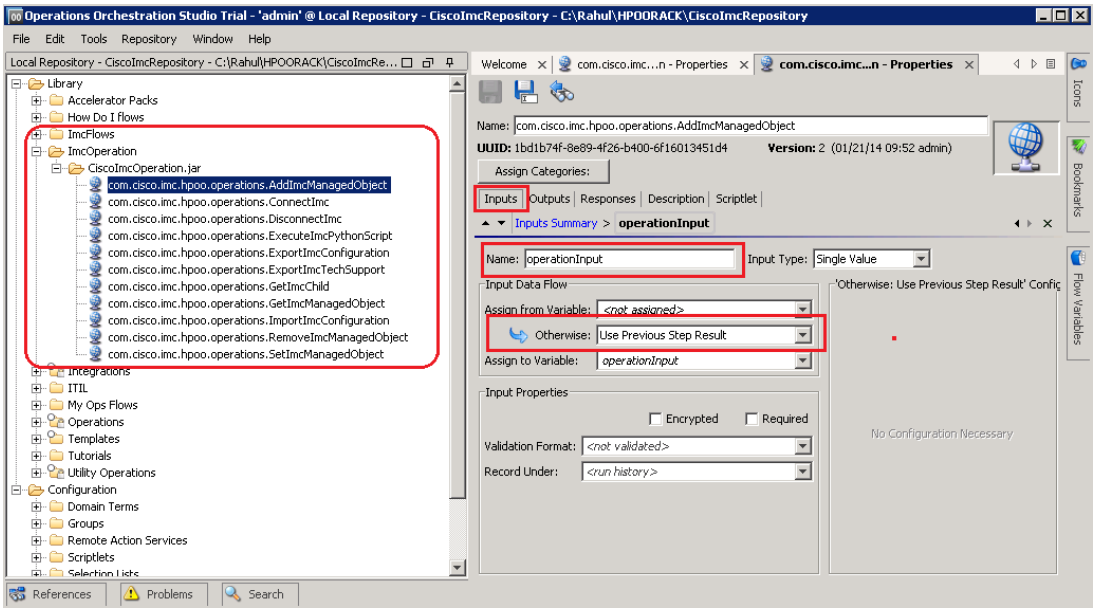

Figure 13: Create Operation from RAS Apply Changes

10. Repeat Step 9 for all operations except ConnectUcs and save the changes.

Note: User would have to publish the data from Studio to Central repository for making the new Operations available in Central repository as well.

4.5 Publishing from Studio to Central Repository

Studio repository provides the development environment for creating/modifying flows. Studio repository data can be published to Central repository, as Central repository is easily accessible on the web and a user with appropriate privileges can easily use these flows from the HPOO web interface.

The following steps need to be performed for publishing data from Studio repository to Central repository:

1. From the Studio client, select menu **Repository -> Set Target Repository… -> <select one of the Central Repositories in the menu>**.

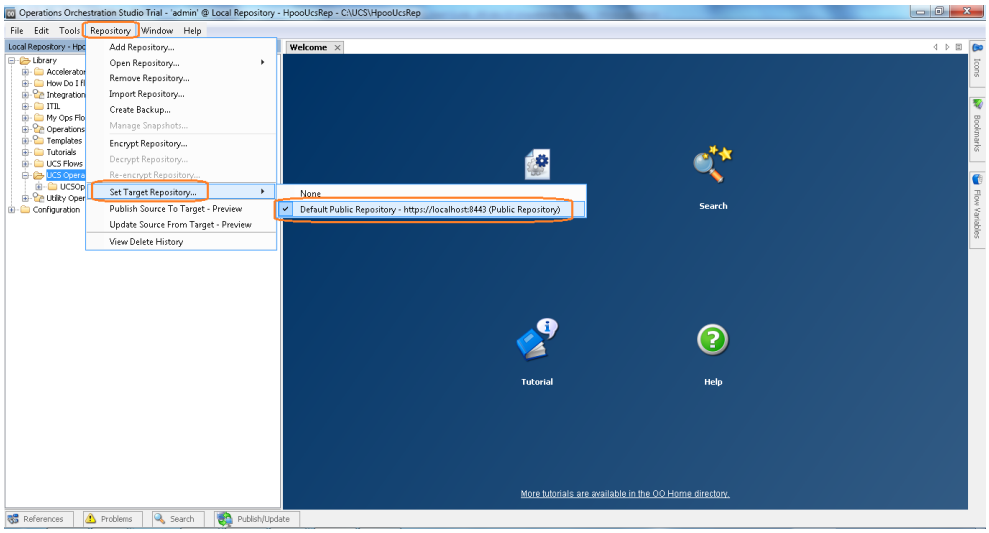

Figure 14: Set Target Repository

2. In the menu select **Repository -> Publish Source To Target - Preview**.

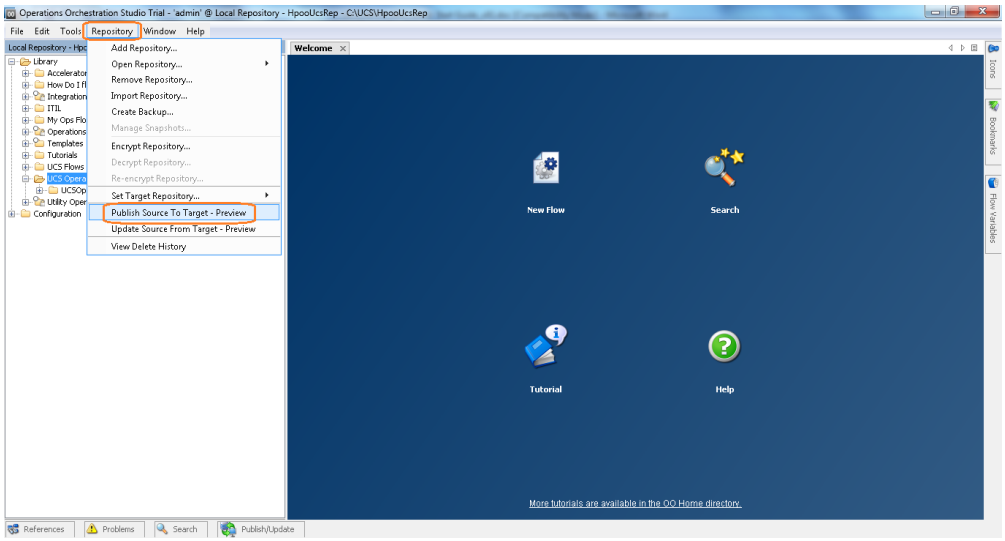

Figure 15: Publish Source To Target

3. Verify the changes shown in the Publish/Update tab and click "**Apply**". This publishes studio repository data to central repository.

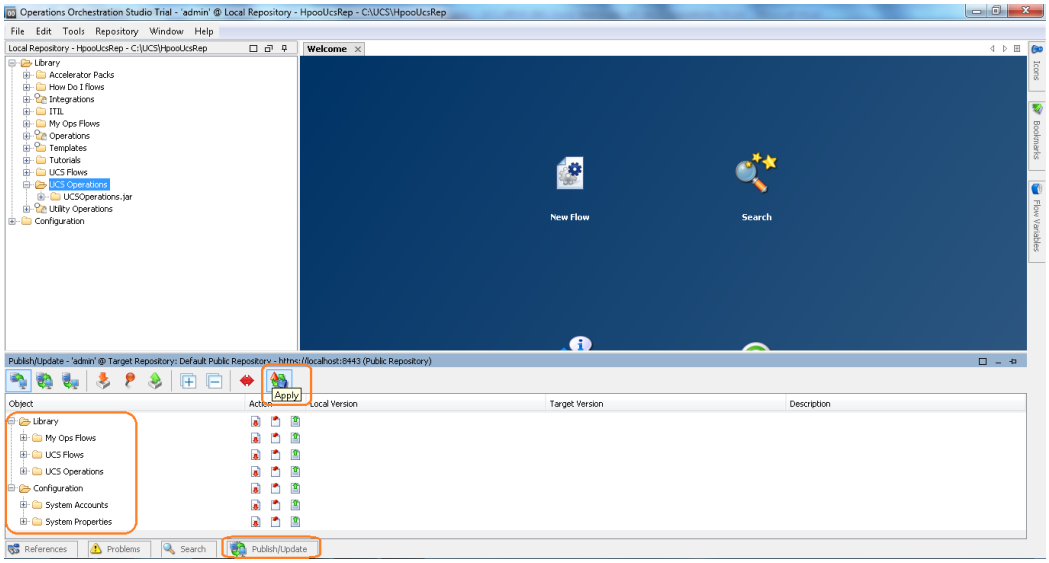

Figure 16: Publish Preview

5 Deleting IMC Operations/Flows from HPOO Repositories

This section explains how to delete already existing Operations/Flows from Central and Studio repositories.

5.1 Deleting from Central Repository

To delete any existing Operation/Flow from Central repository, perform the following steps:

1. From the Central repository, select the Operations/Flows to be removed. Right-click on the selection and click **Delete** button. The deleted Operations/Flows are shown in the **"My Changes/Checkouts"** panel.
2. From the **"My Changes/Checkouts"** panel, either click **"Check In..."** for individually deleted components or **"Check In Tree…"** to complete the delete operation.

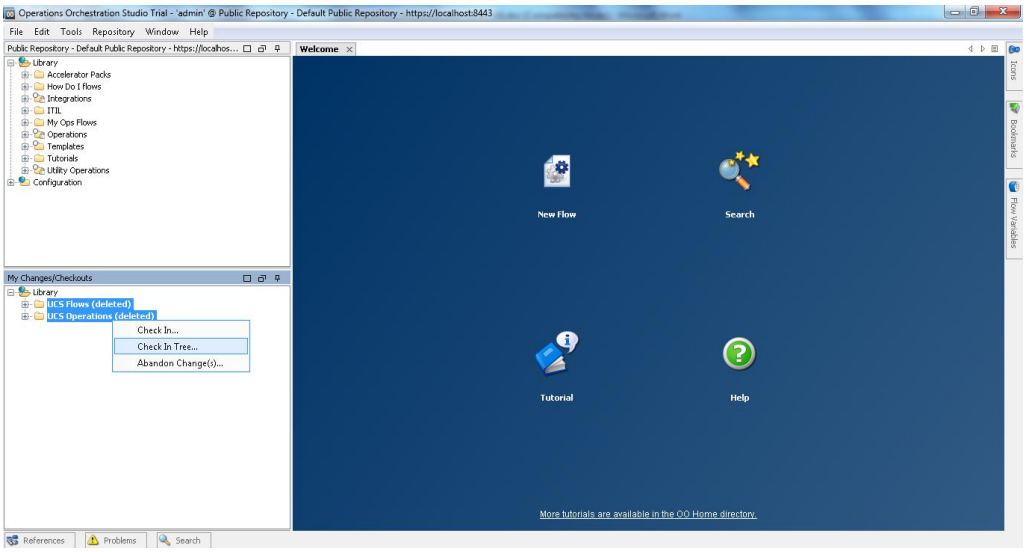

Figure 17: Delete from Central

5.2 Deleting from Studio Repository

In Studio repository, select the Operations/Flows to be removed. Right-click and then click the **Delete** option.

6 Running IMC Flows

When the flows are available in Central repository, user can run these flows from the Central web GUI. Perform the following steps to run these flows:

1. Launch HPOO central web GUI from "*https://<host>:8443/PAS*", where, host is the hostname or IP address of the machine on which HPOO central is installed.

2. In the Flow Library tab, **Selection Tree** click a flow to open it.

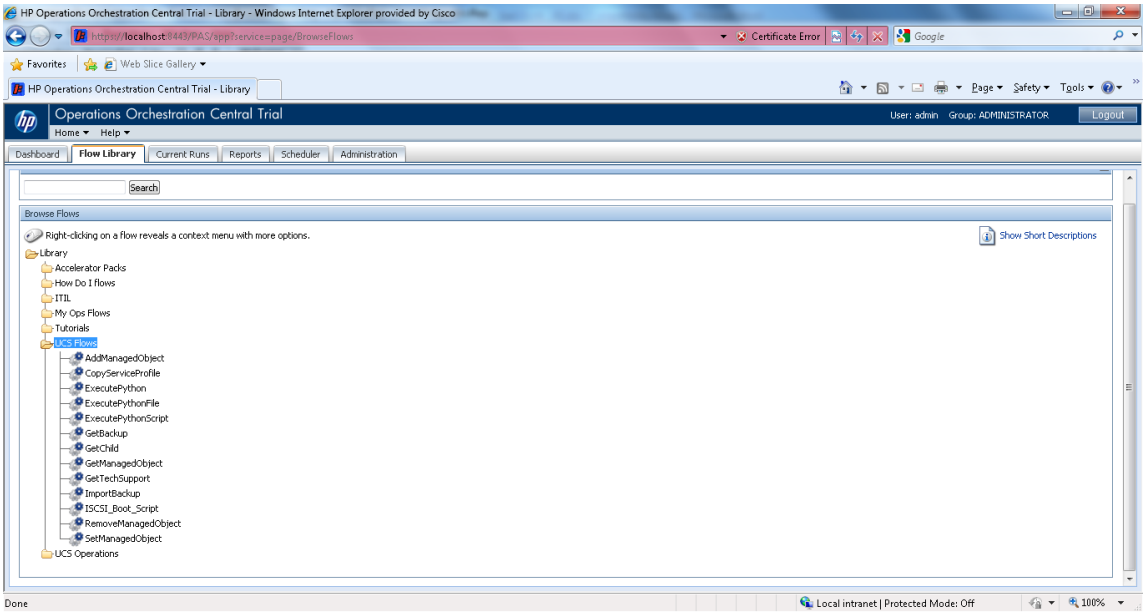

Figure 18: Central Flows

3. Click **Run All** button for running the selected flow.

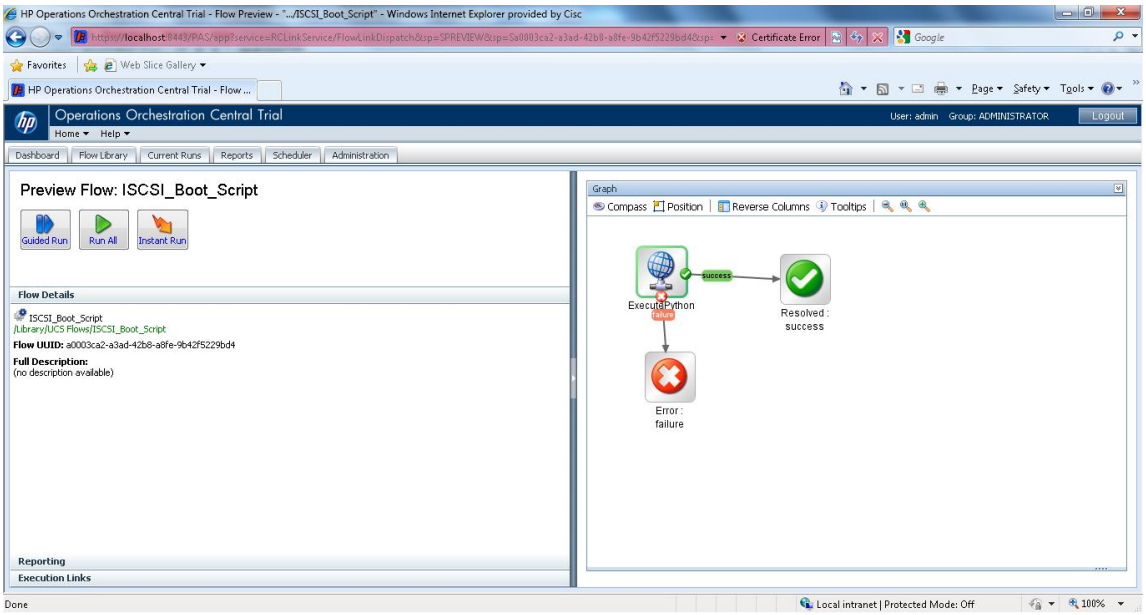

Figure 19: Running Central Flows

4. A pop-up window will prompt the user to provide inputs to the flow parameters. These are required for running a flow. The mandatory parameters are marked with an asterisk (*).

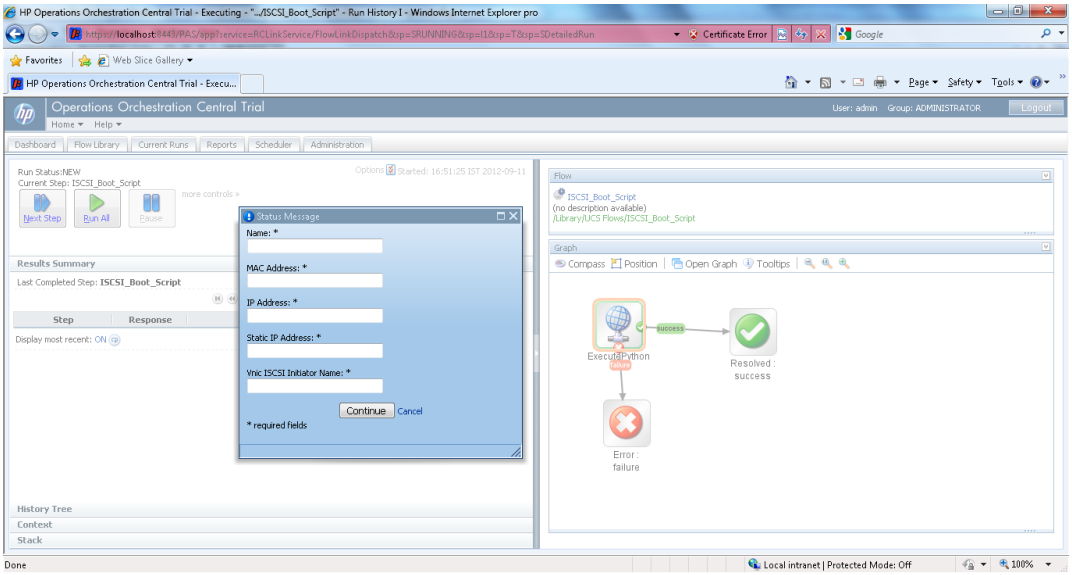

Figure 20: Central Flow Parameters

5. Runtime execution results of the operations in a flow are displayed on the Results Summary window. In case any operation or sub-flow fails, the main flow also fails.

   If all the operations and sub-flows complete successfully, main flow also completes successfully.

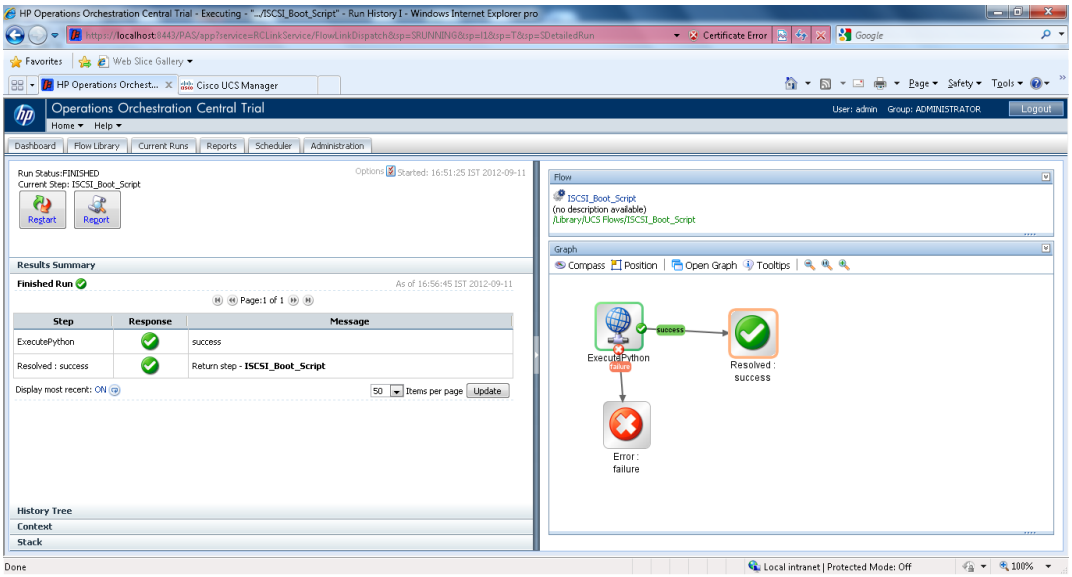

Figure 21: Central Flow Result

7 Cisco IMC Operations

This RAS adds a new group of Operations (Cisco IMC) in the Repository pane of HPOO Studio. It consists of the below listed Operations.

7.1 ConnectImc

This operation establishes a new connection with UCS CSeries.

Operation Data Inputs:

1. hostname - Mandatory: Host Name for UCS CSeries.
2. username - Mandatory: Login User Name.
3. password - Mandatory: Login Password.
4. port - Optional: Http/Https port.
5. noSsl - Optional: Use secure(Https) or non-secure(http) connection.

Operation Control Inputs:

1. operationInput : Contains output of previous operation.
2. defaultImc : Contains connected IMC info. This field uses a flow variable. It contains list of all connected IMC servers.

Responses:

1. success - Operation successfully completed.
2. failure - Operation failed to complete.

Results:

1. operationOutput - Contains output of current operation (to be used by next operation as operationInput).
2. displayOutput - Contains data to be displayed.
3. defaultImc : Contains connected IMC server info. This result can be used to update DefaultImc flow variable which is a list of all connected IMC servers.

7.2 DisconnectImc

Disconnects This Operation disconnects from IMC servers.

Operation Control Inputs:

1. operationInput : Contains output of previous operation.
2. defaultImc : Contains connected IMC info. This field uses a flow variable. It contains list of all connected IMC servers.
3. imc : Comma separated list of connected IMC names on which this operation wants to work on.

Responses:

1. success - Operation successfully completed.
2. failure - Operation failed to complete.

Results:

1. operationOutput - Contains output of current operation (to be used by next operation as operationInput).
2. displayOutput - Contains data to be displayed.
3. defaultImc : Contains connected IMC server info. This result can be used to update DefaultImc flow variable which is a list of all connected IMC servers.

7.3 GetImcManagedObject

This Operation gets a Managed Object from IMC server.

Operation Data Inputs:

1. classId - Optional: ClassId/Name of managed object to get.
2. args - Optional: semicolon (;) separated list of key/value pairs(key=value), that are used as filters for selecting specific Managed Objects. The key should be a valid property of the Managed Object to be retrieved.
3. hierarchy - Optional: Explores hierarchy if true, else returns single level Managed Objects.

Operation Control Inputs:

1. operationInput : Contains output of previous operation.
2. defaultImc : Contains connected IMC info. This field uses a flow variable. It contains list of all connected IMC servers.
3. imc : Comma separated list of connected IMC names on which this operation wants to work on.

Responses:

1. success - Operation successfully completed.
2. failure - Operation failed to complete.

Results:

1. operationOutput - Contains output of current operation (to be used by next operation as operationInput).
2. displayOutput - Contains data to be displayed.
3. defaultImc : Contains connected IMC server info. This result can be used to update DefaultImc flow variable which is a list of all connected IMC servers.

7.4 SetImcManagedObject

This Operation modifies a Managed Object in IMC Server.

Operation Data Inputs:

1. classId - Optional: ClassId/Name of managed object to set.
2. args - Mandatory: semicolon (;) separated list of key/value pairs(key=value), that needs to be updated. The key should be a valid updatable property of the Managed Object to be modified.

Operation Control Inputs:

1. operationInput : Contains output of previous operation.
2. defaultImc : Contains connected IMC info. This field uses a flow variable. It contains list of all connected IMC servers.
3. imc : Comma separated list of connected IMC names on which this operation wants to work on.

Responses:

1. success - Operation successfully completed.
2. failure - Operation failed to complete.

Results:

1. operationOutput - Contains output of current operation (to be used by next operation as operationInput).
2. displayOutput - Contains data to be displayed.
3. defaultImc : Contains connected IMC server info. This result can be used to update DefaultImc flow variable which is a list of all connected IMC servers.

7.5 AddImcManagedObject

This Operation adds a Managed Object to IMC server.

Operation Data Inputs:

1. classId - Mandatory: ClassId/Name of managed object to add.
2. args - Optional: semicolon (;) separated list of key/value pairs(key=value), that are needed by the Managed Object to be added. The key should be any non-readonly property of the Managed Object to be added.

Operation Control Inputs:

1. operationInput : Contains output of previous operation.
2. defaultImc : Contains connected IMC info. This field uses a flow variable. It contains list of all connected IMC servers.
3. imc : Comma separated list of connected IMC names on which this operation wants to work on.

Responses:

1. success - Operation successfully completed.
2. failure - Operation failed to complete.

Results:

1. operationOutput - Contains output of current operation (to be used by next operation as operationInput).
2. displayOutput - Contains data to be displayed.
3. defaultImc : Contains connected IMC server info. This result can be used to update DefaultImc flow variable which is a list of all connected IMC servers.

7.6 RemoveImcManagedObject

This Operation removes a Managed Object from IMC server.

Operation Data Inputs:

1. classId - Optional: ClassId/Name of managed object to remove.
2. args - Optional: semicolon (;) separated list of key/value pairs(key=value), that are needed by the Managed Object to be added. The key can only be Dn property only.

Operation Control Inputs:

1. operationInput : Contains output of previous operation.
2. defaultImc : Contains connected IMC info. This field uses a flow variable. It contains list of all connected IMC servers.
3. imc : Comma separated list of connected IMC names on which this operation wants to work on.

Responses:

1. success - Operation successfully completed.
2. failure - Operation failed to complete.

Results:

1. operationOutput - Contains output of current operation (to be used by next operation as operationInput).
2. displayOutput - Contains data to be displayed.
3. defaultImc : Contains connected IMC server info. This result can be used to update DefaultImc flow variable which is a list of all connected IMC servers.

7.7 ExportImcTechSupport

This Operation gets TechSupport information from IMC and saves it to a file.

Operation Data Inputs:

1. remoteHost - Mandatory: Remote host where user needs to download technical support data.
2. userName - Mandatory: Username to login to remote host.
3. password - Mandatory: Password to login to remote host.
4. remoteFile - Mandatory: Filename with full path where user needs to download the technical support data file. Note: Should be a tar.gz file.
5. proto - Mandatory: Protocol used for transferring the file to remote.
6. timeoutSec - Optional: Waits for specified maximum time in sec for the tech support file to generate else exit.

Operation Control Inputs:

1. operationInput : Contains output of previous operation.
2. defaultImc : Contains connected IMC info. This field uses a flow variable. It contains list of all connected IMC servers.
3. imc : Comma separated list of connected IMC names on which this operation wants to work on.

Responses:

1. success - Operation successfully completed.
2. failure - Operation failed to complete.

Results:

1. operationOutput - Contains output of current operation (to be used by next operation as operationInput).
2. displayOutput - Contains data to be displayed.
3. defaultImc : Contains connected IMC server info. This result can be used to update DefaultImc flow variable which is a list of all connected IMC servers.

7.8 ExportImcConfiguration

This Operation gets backup of the IMC configuration and saves it to a file.

Operation Data Inputs:

1. remoteHost - Mandatory: Remote host where user needs to download technical support data.
2. userName - Mandatory: Username to login to remote host.
3. password - Mandatory: Password to login to remote host.
4. remoteFile - Mandatory: Filename with full path where user needs to download the Imc configuration file. Note: Should be a .xml file.
5. proto - Mandatory: Protocol used for transferring the file to remote.
6. timeoutSec - Optional: Waits for specified maximum time in sec for the tech support file to generate else exit.

Operation Control Inputs:

1. operationInput : Contains output of previous operation.
2. defaultImc : Contains connected IMC info. This field uses a flow variable. It contains list of all connected IMC servers.
3. imc : Comma separated list of connected IMC names on which this operation wants to work on.

Responses:

1. success - Operation successfully completed.
2. failure - Operation failed to complete.

Results:

1. operationOutput - Contains output of current operation (to be used by next operation as operationInput).
2. displayOutput - Contains data to be displayed.
3. defaultImc : Contains connected IMC server info. This result can be used to update DefaultImc flow variable which is a list of all connected IMC servers.

7.9 ImportImcConfiguration

This Operation imports IMC backup configuration from a file to IMC server.

Operation Data Inputs:

1. remoteHost - Mandatory: Remote host where user needs to download technical support data.
2. userName - Mandatory: Username to login to remote host.
3. password - Mandatory: Password to login to remote host.
4. remoteFile - Mandatory: Filename with full path where resides the Imc configuration file. Note: Should be a .xml file.
5. proto - Mandatory: Protocol used for transferring the file to remote.

Operation Control Inputs:

1. operationInput : Contains output of previous operation.
2. defaultImc : Contains connected IMC info. This field uses a flow variable. It contains list of all connected IMC servers.
3. imc : Comma separated list of connected IMC names on which this operation wants to work on.

Responses:

1. success - Operation successfully completed.
2. failure - Operation failed to complete.

Results:

1. operationOutput - Contains output of current operation (to be used by next operation as operationInput).
2. displayOutput - Contains data to be displayed.
3. defaultImc : Contains connected IMC server info. This result can be used to update DefaultImc flow variable which is a list of all connected IMC servers.

7.10 GetImcChild

This Operation gets all children of any managed object.

Operation Data Inputs:

1. classId - Optional: class id of any particular child. If empty then all children are extracted.
2. hierarchy - Optional: Boolean flag to explore full hierarchy. Default value is false.

Operation Control Inputs:

1. operationInput : Contains output of previous operation.
2. defaultImc : Contains connected IMC info. This field uses a flow variable. It contains list of all connected IMC servers.
3. imc : Comma separated list of connected IMC names on which this operation wants to work on.

Responses:

1. success - Operation successfully completed.
2. failure - Operation failed to complete.

Results:

1. operationOutput - Contains output of current operation (to be used by next operation as operationInput).
2. displayOutput - Contains data to be displayed.
3. defaultImc : Contains connected IMC server info. This result can be used to update DefaultImc flow variable which is a list of all connected IMC servers.

8 IMC Terminologies

There are few IMC Terminologies that user must be aware of before writing the python scripts. These terminologies are described in this section.

8.1.1 Management Information Model

All the physical and logical components that comprise IMC are represented in a hierarchical Management Information Model (MIM), referred to as the Management Information Tree (MIT). Each node in the tree represents a Managed Object (MO), uniquely identified by its Distinguished Name. (DN)

The figure below illustrates a sample (partial) MIT for three rackunits.

```
Tree (topRoot)                Distinguished Name
|— sys                        sys
    |— rack-unit-1              sys/rack-unit-1
    |— rack-unit-2              sys/rack-unit-2
    |— rack-unit-3              sys/rack-unit-3
         |— adaptor-1           sys/rack-unit-3/adaptor-1
```

8.1.1.1 Managed Objects

Managed Objects (MO) are abstractions of Cisco IMC resources, such as rack-mounted servers, cpu, memory and network adaptors. Managed Objects represent any physical or logical entity that is configured / managed in the Cisco IMC MIT. For example, physical entities such as rack unit, I/O cards, Processors and logical entities such as User roles are represented as Managed Objects.

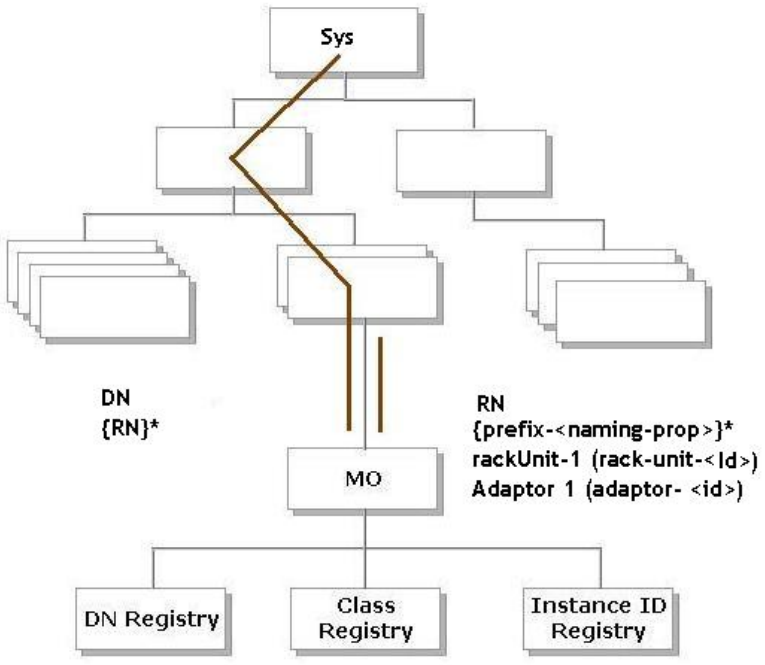

Figure 22: Management Information Tree

Every Managed Object is uniquely identified in the tree with its Distinguished Name (Dn) and can be uniquely identified within the context of its parent with its Relative Name (Rn). The Dn identifies the place of the MO in the MIT. A Dn is a concatenation of all the relative names starting from the root to the MO itself. Essentially, Dn = [Rn]/[Rn]/[Rn]/…/[Rn].

In the example below, Dn provides a fully qualified name for adaptor-1 in the model.

< dn = “sys/rack-unit-2/adaptor-1” />

The above written Dn is composed of the following Rn:

topSystem MO: rn="sys"
rack unit MO: rn="rack-unit-<id>"
adaptorUnit MO: rn="adaptor-<id>"

A Relative Name (Rn) *may* have the value of one or more of the MO’s properties embedded in it. This allows in differentiating multiple MOs of the same type within the context of the parent. Any properties that form part of the Rn as described earlier are referred to as Naming properties.

For instance, multiple adaptor MOs reside under a rack unit MO. The adaptor MO contains the adaptor identifier as part of its Rn (adaptor-[id]), thereby uniquely identifying each adaptor MO in the context of a rack unit.

8.1.1.2 References to Managed Objects

The contents of the Managed Objects are referred to during the operation of IMC. Some of the MOs are referred to implicitly or as part of deployment of another MO.

The different types of references can be classified as shown below:

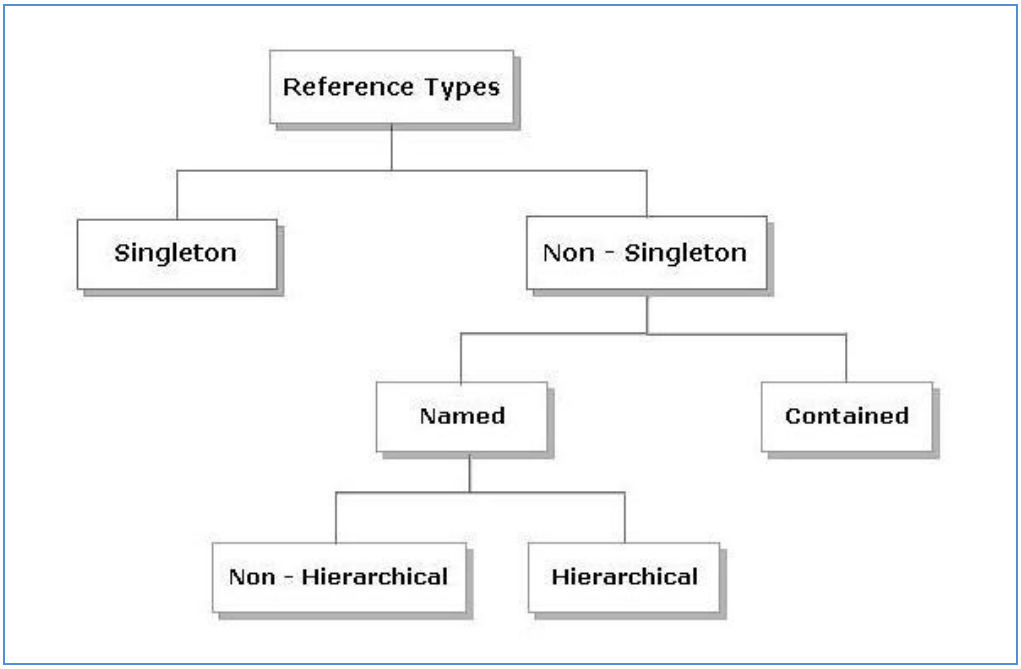

Figure 23: References to Managed Objects

A singleton MO type is found at most once in the entire MIT and is typically referred to implicitly.

Non-Singleton MO type may be instantiated one or more times in the MIT. In many cases, when an MO refers to another, the reference is made by name. Depending on the type of the referenced MO, the resolution may be hierarchical.

8.1.1.3 Properties of Managed Objects

Properties of Managed Objects can be classified as Configuration or Operational.

Configuration properties may be classified as:

- Naming properties: Form part of the Rn. Needs to be specified only during MO creation and cannot be modified later.
- Create-Only properties: May be specified only during MO creation and cannot be modified later. If the property is not specified, a default value is assumed.
- Read / Write properties: May be specified during MO creation and can also be modified subsequently.

Operational properties indicate the current status of the MO / system and are hence read-only.

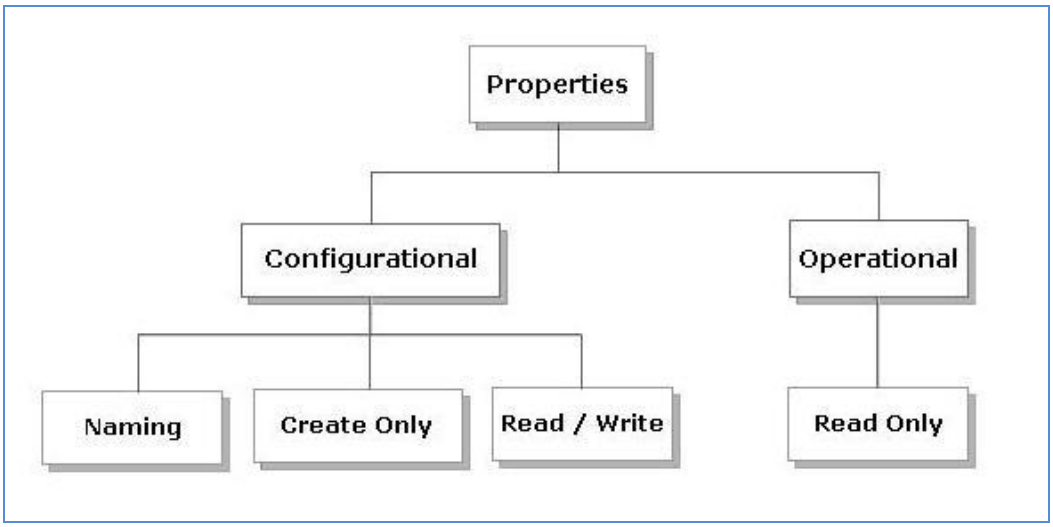

Figure 24: Property Types of Managed Objects

The table below lists the examples of the various property types.

Property Type	Example
Naming	id in computeRackUnit (Rack Unit MO)
Read / Write	usrLbl in computeRackUnit
Read-Only	totalMemory in computeRackUnit

8.1.1.4 Methods

Methods are Cisco IMC XML APIs, used to manage and monitor the system. There are methods supported for:

- aaaLogin — Initial method for logging in.
- aaaRefresh — Refreshes the current authentication cookie.
- aaaLogout — Exits the current session and deactivates the corresponding authentication cookie.
- configResolveDn— Retrieves objects by DN.
- configResolveClass — Retrieves objects of a given class.
- configResolveChildren — Retrieves the child objects of an object.
- configResolveParent — Retrieves the parent object of an object.
- configConfMo — Affects single managed object.
- eventSubscribe — To register for events.

9 Execute Imc Python Script

This method provides the python interface to the CISCO plug-in.

9.1 callFlow function

The starting point of Execution of python script in HPOO is callFlow function, hence all the script must be written in callFlow function. Syntax of callFlow function is described below:

Script Example:-

```
Line1   def callFlow(inMap, outDisplay, outPublish):
Line2
Line3           handleList = []
Line4           scriptSuccess = False
Line5
Line6
Line7           try:
Line8                   handle = Connect({"HostName":"${hostname}",
"UserName":"username", "Password":"password"})
Line9                   if(handle.success):
Line10                          scriptSuccess = True
Line11                          handleList.append(handle)
Line12                  outDisplay.add(OperationOutput(handle.success, handle.message,
None))
Line13
Line14                  # Get ComputeRackUnit
Line15                  rack = handle.GetManagedObject(None,'computeRackUnit')
Line16                  if rack : outDisplay.add(rack)
Line17
Line18                  # Update ComputeRackUnit Description
Line19                  updateRack =
handle.SetManagedObject(rack,None,{"usrLbl":"Testing ExecutePython"})
Line20                  if updateRack : outDisplay.add(updateRack)
Line21
Line22                  # Get ComputeBoard child under rack with hierarchy
Line23                  rackChild = handle.GetUcsCSeriesChild(rack, {"classId":
"computeBoard" , "hierarchy" :"True"})
Line24                  if rackChild : outDisplay.add(rackChild)
Line25
Line26                  # Add LsBootEFI
Line27                  bootDef = handle.GetManagedObject(None,'lsBootDef')
Line28                  if bootDef : outDisplay.add(bootDef)
Line29                  bootEfi = handle.AddManagedObject(bootDef, classId="LsBootEfi",
params=None)
Line30                  if bootEfi : outDisplay.add(bootEfi)
Line31
Line32                  # Remove LsBootEFI
Line33                  bootEfi = handle.GetManagedObject(None,'LsBootEfi')
Line34                  if bootEfi : outDisplay.add(bootEfi)
Line35                  removeBootEfi = handle.RemoveManagedObject(bootEfi)
Line36                  if removeBootEfi : outDisplay.add(removeBootEfi)
Line37
Line38          except Exception, err:
Line39                  outDisplay.add(OperationOutput(False, str(err), None))
Line40                  scriptSuccess = False
Line41
Line42          disconnectOutput = Disconnect(handleList)
Line43          if(scriptSuccess):
Line44                  scriptSuccess = disconnectOutput.isSuccess()
Line45          outDisplay.add(disconnectOutput)
```

```
Line46
Line47          return scriptSuccess
```

The callFlow function must accept three arguments:-

1. **inMap:** It contains any output objects from previous HPOO operations.
2. **outDisplay:** This is a list of outputs that user wants to display as the output of the ExecutePython operation (as added in Line8, Line12 and Line20.
3. **outPublish:** This is a list of outputs that we want to send to next HPOO operation in the flow.

The callFlow function must return a Boolean flag that indicates the success or failure of the script.

9.1.1 Connect Operation

Connect Operation performs a Login to a specified IMC, and returns a handle, that can be used to perform various operations in the script. It takes a dictionary object as argument. The key value pairs that should be passed to connect operation are as below (Please see Line4 of Script Example shown in section 6.12):-

1. **HostName:** HostName/IP Address of the IMC. This parameter is mandatory.
2. **UserName:** UserID of IMC to login. This parameter is mandatory.
3. **Password:** Password of IMc to login. This parameter is mandatory.
4. **NoSsl:** true/false value for specifying if Login is through secured connection. This parameter is optional. Default value is false (secured connection).
5. **Port:** port to which you want to connect. This parameter is optional. Default values are 80 for non-secure and 443 for secure connection.

9.1.2 Disconnect Operation

Disconnect operation takes a list of handles to Disconnect from multiple handles (Please see Line17 of Script Example shown in section 6.12).

9.1.3 OperationOutput Object

Every operation (except Connect, which returns a handle) returns OperationOutput object. An OperationOutput object contains a success flag, a message and a list of objects as a result of operation execution on IMC (please see Line8 and Line14 of Script Example shown in section 6.12).

9.1.4 Handle Object

Handle object represents a live connection to the IMC. Connect operation creates a handle object, and Disconnect operation destroys a handle object.

All the operations on the IMC are exposed through handle object. Following sub-sections describe all operations exposed by handle object.

9.1.4.1 Generic Operations

There are four generic operations exposed through handle.

1. **AddImcManagedObject:** Adds a specified Managed Object in IMC.
   Its parameters are:
   - i) **prevOutput:-** Previous OperationOutput that acts as a parent of the Managed Object to be added.
   - ii) **classId:-** Class Id or noun of Managed Object to be added.
   - iii) **params:-** a dictionary (key/value pairs) of arguments to be passed while adding Managed Object. Default value is “None”.

2. **GetImcManagedObject:** Retrieves Managed Objects from IMC as per the inputs provided.
   Its parameters are:
   - i) **prevOutput:-** Previous OperationOutput that acts as a parent of the Managed Object to be retrieved.
   - ii) **classId:-** Class Id or noun of Managed Object to be retrieved.
   - iii) **params:-** a dictionary (key/value pairs) of arguments to be passed as filters while retrieving Managed Object. Default value is “None”.
   - iv) **inHierarchy:-** A boolean flag for getting full hierarchy (all children) of the Managed Object. Default value is “False”.

3. **SetImcManagedObject:** Modifies an existing Managed Object in IMC.
   Its parameters are:
   - i) **prevOutput:-** Previous OperationOutput that acts as the Managed Object to be modified.
   - ii) **classId:-** Class Id or noun of Managed Object to be modified. Default value is “None”.
   - iii) **params:-** a dictionary (key/value pairs) of arguments to be passed as attributes to be modified of Managed Object. Default value is None.

4. **RemoveManagedObject:** Removes an existing Managed Object from IMC
   Its parameters are:
   - i) **prevOutput:-** Previous OperationOutput that acts as the Managed Object to be removed.
   - ii) **classId:-** Class Id or noun of Managed Object to be removed. Default value is “None”.
   - iii) **params:-** a dictionary (key/value pairs) of arguments to be passed as filters for identifying which existing Managed Object to remove. Currently only “Dn” can be given as argument. Default value is “None”.

9.1.4.2 Custom Operations

There are few utility operations that user may want to perform.

There are 4 customer operations exposed through handle.

1. **ExportImcTechSupport:** Generate and retrieve Tech Support files from IMC.
   Its parameters are:
   i) **params:-** a dictionary (key/value pairs) of arguments to be passed while getting backup. Please see Operation Data Inputs of section 6.7 for description of arguments of ExportImcTechSupport.

3. **ExportImcConfiguration:** Gets the backup of IMC, and saves it to specified xml file.
   Its parameters are:
   i) **params:-** a dictionary (key/value pairs) of arguments to be passed while importing backup. Please see Operation Data Inputs of section 6.8 for description of arguments of ExportImcConfiguration.

4. **ImportImcConfiguration:** Imports the backup (xml file generated by ExportImcConfiguration) into IMC.
   Its parameters are:
   i) **params:-** a dictionary (key/value pairs) of arguments to be passed while getting Tech Support files from IMC. Please see Operation Data Inputs of section 6.9 for description of arguments of ImportImcConfiguration.

5. **GetImcChild:** This Operation gets all children of any Managed Object.
   Its parameters are:
   i) **prevOutput:-** Previous OperationOutput that acts as the Managed Object from which all children needs to be retrieved.
   ii) **params:-** a dictionary (key/value pairs) of arguments to be passed as filters for identifying which existing child Managed Objects to retrieve from specified Managed Object. Please see Operation Data Inputs of section 6.10 for description of arguments of GetImcChild.

9.1.4.3 External Methods

If user wants to perform tasks that are not covered by existing operations, then HPOO also exposes XML API methods directly. These can be used from within the ExecuteImcPythonScript Operation. (Please see References for IMC XML API Guide). Please see Python SDK for usage of External Methods.

External Method Script Example:-

```
Line1 def callFlow(inMap, outDisplay, outPublish):
Line2
Line3   handleList = []
Line4   scriptSuccess = False
Line5
Line6   try:
Line7           handle = Connect({"HostName":"${hostname}", "UserName":"${username}",
"Password":"${password}"})
Line8           if(handle.success):
Line9                   scriptSuccess = True
Line10                  handleList.append(handle)
Line11          outDisplay.add(OperationOutput(handle.success, handle.message, None))
Line12
Line13          crDn = handle.ConfigResolveClass('computeRackUnit')
Line14          outDisplay.add(toHpoo(crDn))
Line15
Line16  except Exception, err:
Line17          outDisplay.add(OperationOutput(False, str(err), None))
Line18          scriptSuccess = False
Line19
Line20  disconnectOutput = Disconnect(handleList)
Line21  if(scriptSuccess):
Line22          scriptSuccess = disconnectOutput.isSuccess()
Line23  outDisplay.add(disconnectOutput)
Line24
Line25  return scriptSuccess
```

List of External Methods exposed by handle:-

1. AaaKeepAlive()
2. AaaLogin(inName, inPassword)
3. AaaLogout()
4. AaaRefresh(inName, inPassword)
5. ConfigConfMo(dn, inConfig, inHierarchical=YesOrNo.FALSE)
6. ConfigResolveChildren(classId, inDn, inFilter, inHierarchical=YesOrNo.FALSE)
7. ConfigResolveClass(classId, inHierarchical=YesOrNo.FALSE)
8. ConfigResolveDn(dn, inHierarchical=YesOrNo.FALSE)
9. ConfigResolveParent(dn, inHierarchical=YesOrNo.FALSE)
10. EventSubscribe()
11. EventUnsubscribe()

9.1.5 Utility Methods

There are few utility methods provided in the HPOO python scripts that can be used for convenience.

Utility Methods Script Example:-

```
Line1   def callFlow(inMap, outDisplay, outPublish):
Line2
Line3           handleList = []
Line4           scriptSuccess = False
Line5           pipelineObjects = getPipelineObjects(inMap, handleList)
Line6
Line7           for handle in handleList:
Line8                   try:
Line9
Line10                          fan = handle.GetManagedObject(getPipeline(handle,
pipelineObjects), 'EquipmentFan')
Line11                          if fan :
Line12                                  outDisplay.add(fan)
Line13                                  scriptSuccess = True
Line14
Line15                  except Exception, err:
Line16                          outDisplay.add(OperationOutput(False, str(err), None))
Line17                          scriptSuccess = False
Line18
Line19          updateDefaultUcsList(inMap, handleList)
Line20
Line21          return scriptSuccess
```

There are four utility methods that can be used in HPOO python scripts:-

1. **getPipelineObjects:** This method is used when user want ExecutePython Operation to be linked with other previous operations in HPOO flow. getPipelineObjects extracts the objects of previous operation and provides it to the script. It also extracts handles from previous operation and puts them into a List (Please see Line05 in above example).

2. **getPipeline:** This method is used in conjunction with getPipelineObjects. Output of getPipelineObjects could contain objects from more than one UCS (handle), getPipeline method extracts objects of one handle from it (Please see Line10 in above example).

3. **toHpoo:** This method is used to extract objects from an external method and put it into an OperationOuput object, as HPOO understands OperationsOutput object. (Please see Line14 in External Method Script example).

4. **updateDefaultUcsList:** This method is used to update the DefaultUcsList in the HPOO flow. (Please see Line19 in above example).

References

1. For more information about Cisco UCS CSeries, visit
   http://www.cisco.com/en/US/products/ps10493/index.html

2. For more information on Hewlett-Packard Operations Orchestration, its installation, or features and functionalities refer to the Hewlett-Packard support website http://support.openview.hp.com/selfsolve/manuals. If you have already installed HPOO then you can find documentation about studio and central in <HPOO Home>/studio/docs, and <HPOO Home>/central/docs directories.

3. For More information on UCS Series XML API, please see
   http://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/c/sw/api/b_cimc_api_book.html

4. Cisco UCS CSeries CLI Configuration Guide
   http://www.cisco.com/en/US/docs/unified_computing/ucs/c/sw/cli/config/guide/131/b_Cisco_UCS_C-Series_CLI_Configuration_Guide_131.html