Cisco IMC Python SDK 0.7.1 User Guide

Nov 20, 2014

Table of Contents

1 Overview

Cisco IMC Python SDK is a python module which helps automate all aspects of Cisco IMC management servers i.e. both C-Series and E-Series.

Bulk of the Cisco IMC Python SDK work on the IMC Manager's Management Information Tree (MIT), performing create, modify or delete actions on the Managed Objects (MO) in the tree. The next chapter provides an overview of the Cisco IMC Management Information Model (MIM).

2 Management Information Model

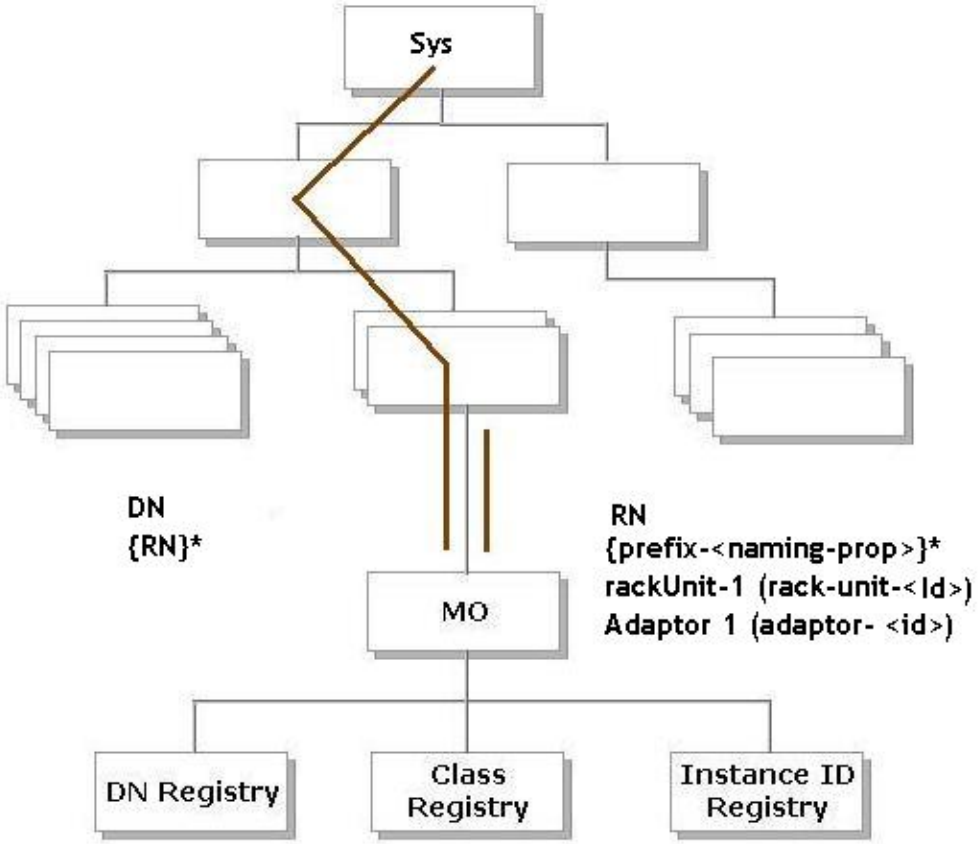
All the physical and logical components that comprise IMC are represented in a hierarchical Management Information Model (MIM), referred to as the Management Information Tree (MIT). Each node in the tree represents a Managed Object (MO), uniquely identified by its Distinguished Name. (DN)

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
```

2.1 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.



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>"

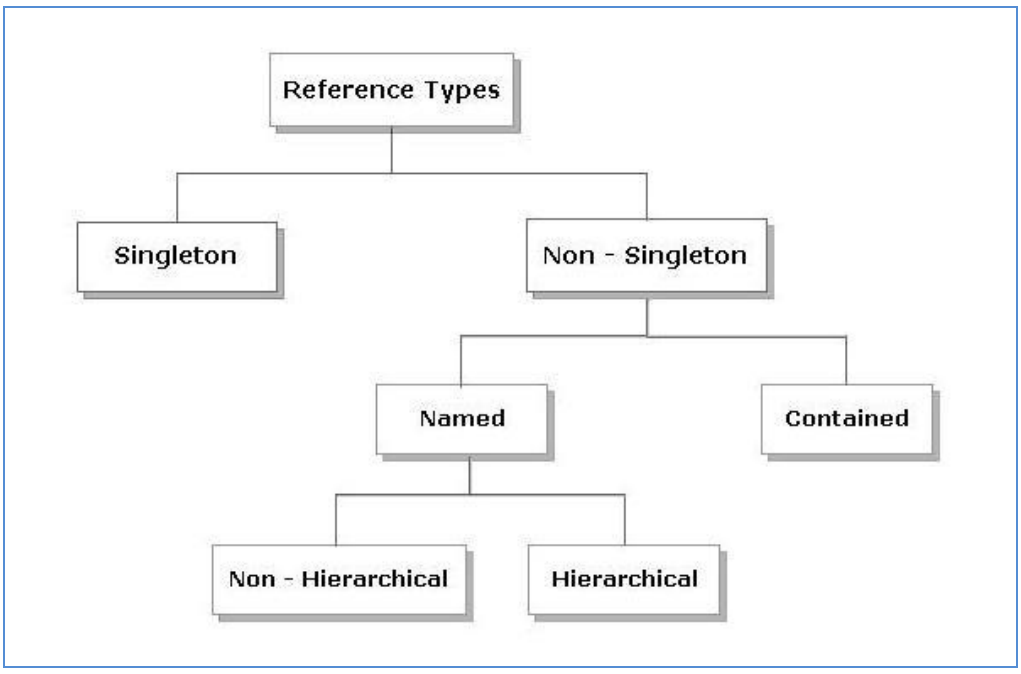
A Relative Name (Rn) *may* have the value of one or more of the MO’s properties embedded in it. This allows in differentiating multiple MOs of the same type within the context of the parent. Any properties that form part of the Rn as described earlier are referred to as Naming properties.

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.

2.2 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:



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.

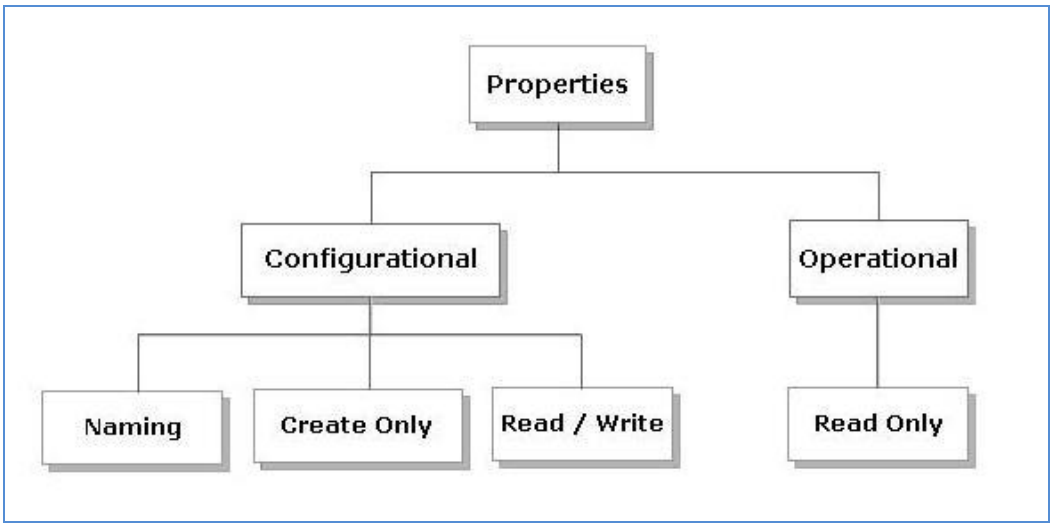
2.3 Properties of Managed Objects

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.



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

2.4 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.

3 Installation

3.1 Pre-Install Checklist

- Ensure you have Python v2.4 or above version of v2.x.
- Supported OS – Cygwin, RHEL 5.x, and other versions of Linux/Unix with Python version v2.4 or above version of v2.x.
- Remove the existing Python IMC Sdk package from the installation directory.
  E.g. Let say, you installed CISCO IMC SDK in Python 2.7.
  The default installation directory will be "C:\Python27\Lib\site-packages\ImcSdk". So first remove ImcSdk directory and then install the fresh ImcSdk.

3.2 Installation Steps

- Download the source distribution of Python IMC SDK.
- Un-tar the package and execute the following from the package.
  - python setup.py build
  - sudo python setup.py install

3.3 Getting Started

1. Launch python.

2. Connect to a IMC system.

```
from ImcSdk import *
handle = ImcHandle()
handle1 = ImcHandle()
handle2 = ImcHandle()

handle.login(<ip or hostname>)
handle1.login(<ip or hostname>, username="<username>", password="<password>")
handle2.login(<ip or hostname>, username="<username>", password="<password>",
nossl=False, port=443, dump_xml=YesOrNo.TRUE)
```

3. Connect to a multiple IMC system.

```
from ImcSdk import *
handle1 = ImcHandle()
handle1.login(<ip or hostname>)
handle2 = ImcHandle()
handle2.login(<ip or hostname>)
```

4. **Disconnect.**

```
handle.logout()
```

A sample script to login and logout is shown below:

```
from ImcSdk import *

if __name__ == "__main__":
       try:
              handle = ImcHandle()
              handle.login("IPAddress","username","password")
              handle.logout()

       except Exception, err:
              print "Exception:", str(err)
              import traceback, sys
              print '-'*60
              traceback.print_exc(file=sys.stdout)
              print '-'*60
```

3.4 What is new in 0.7.1

- Made codebase adhered to standard python coding conventions.
- Refreshed with IMC Schema 2.0(3d)
- XmlParser changed to cElementTree to improve performance.
- Modified URI method to verify correct port.
- Modified Login API to prioritize connection via http or https now depends on both port and noSsl flag rather only noSsl flag.
- Removed keys() function while iterating dictionary to improve execution speed.

Table below lists the standard naming convention followed:

Type	Public	Internal
Classes	CapWords	_CapWords
Exceptions	CapWords	
Functions	lower_with_under()	_lower_with_under()
Global/Class Constants	CAPS_WITH_UNDER	_CAPS_WITH_UNDER
Global/Class Variables	lower_with_under	_lower_with_under
Instance Variables	lower_with_under	_lower_with_under (protected) or __lower_with_under (private)
Method Names	lower_with_under()	_lower_with_under() (protected) or __lower_with_under() (private)
Function/Method Parameters	lower_with_under	
Local Variables	lower_with_under	

NOTE:
For respective parameter names refer individual APIs.
In case of unsure of new names, please try to follow the naming convention in above naming convention table.
For e.g. classId -> class_id , dumpXml -> dump_xml, inMo -> in_mo,
inHierarchical -> in_hierarchical .

Table below lists the new names of API in SDK version 0.7.1:

Module	API Names in Ver 0.6.2	API Names in Ver 0.7.1
ImcHandle	SetDumpXml	set_dumpxml
ImcHandle	UnsetDumpXml	unset_dumpxml
ImcHandle	Login	login
ImcHandle	Logout	logout
ImcHandle	GetImcManagedObject	get_imc_managedobject
ImcHandle	AddImcManagedObject	add_imc_managedobject
ImcHandle	SetImcManagedObject	set_imc_managedobject
ImcHandle	RemoveImcManagedObject	remove_imc_managedobject
ImcUtility	BackupImc	backup_imc
ImcUtility	ImportImcBackup	import_imc_backup
ImcUtility	GetImcTechSupport	get_imc_techsupport
ImcUtility	SyncImcManagedObject	sync_imc_managedobject
ImcUtility	CompareImcManagedObject	compare_imc_managedobject
ImcUtility	UpdateImcFirmwareHUU	update_imc_firmware_huu
ImcUtility	UpdateImcFirmware	update_imc_firmware
ImcUtility	StartImcIODSnapshot	start_imc_iod_snapshot
ImcUtility	StopImcIODSnapshot	stop_imc_iod_snapshot
ImcUtility	ExportImcSession	export_imc_session
ImcUtility	ImportImcSession	import_imc_session

3.5 What is new in 0.6.2

- Fix for the issue “GetImcManagedObject API throws TypeError”

3.6 What is new in 0.6.1

- Fixed the error thrown when using python version 2.7.6
- Errors from communication with UCS are wrapped in `ImcException` and all validation errors are thrown as `ImcValidationException` for custom error handling.
- New sample GetInventory.py added to samples folder

4 API Methods

4.1 get_imc_managedobject

This operation gets a Managed Object from IMC.

```
get_imc_managedobject(in_mo,class_id,params=None,in_hierarchical=False,dump_xml=None)
```

Mandatory Input sets:

(1) class_id.

(or)

(2) "Dn" property in params.

Parameter description:

1. in_mo:
   - Mandatory parameter:
   - It should be **None** unless a username wants to define a parent scope.
   - It is a list containing single MO or multiple MOs.
   - If provided, it acts as a parent for the present operation.

2. class_id:
   - class_id of MO.

3. params:
   - Optional parameter.
   - 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.

4. in_hierarchical:
   - Optional parameter
   - Explores hierarchy if true, else returns Managed Objects at a single level.

5. dump_xml:
   - Optional parameter
   - Outputs the xml requests & responses involved.

4.2 add_imc_managedobject

This operation adds a Managed Object to IMC.

```
add_imc_managedobject(in_mo, class_id, params=None, dump_xml=None)
```

Mandatory Input set:

(1) class_id.

Parameter description:

1. in_mo:
   - Mandatory parameter.
   - It should be **None** unless a username wants to define a parent scope.
   - It is a list containing single MO or multiple MOs.
   - If provided, it acts as a parent for the present operation.

2. class_id:
   - class_id of MO.

3. params:
   - Optional parameter.
   - 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.

4. dump_xml:
   - Optional parameter.
   - Outputs the xml requests & responses involved.

4.3 set_imc_managedobject

This operation modifies a Managed Object in IMC.

```
set_imc_managedobject(in_mo, class_id=None, params=None, dump_xml=None)
```

Parameter description:

1. in_mo:
    - Mandatory parameter.
    - It should be **None** unless a username wants to define a parent scope.
    - It is a list containing single MO or multiple MOs.
    - If provided, it acts as a parent for the present operation.

2. class_id:
    - class_id of MO.

3. params:
    - Optional parameter.
    - 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.

4. dump_xml:
    - Optional parameter.
    - Outputs the xml requests & responses involved.

4.4 remove_imc_managedobject

This operation removes a Managed Object from IMC.

```
remove_imc_managedobject(in_mo, class_id=None, params=None, dump_xml=None)
```

Mandatory Input set:

(1) inMOs containing MO(s) to be deleted.
(or)
(2) class_id with "Dn" property in params.

Parameter description:

1. in_mo:
    - Mandatory parameter.
    - It should be **None** unless a username wants to define a parent scope.
    - It is a list containing single MO or multiple MOs.
    - If provided, it acts as a parent for the present operation.

2. class_id:
    - class_id of MO.

3. params:
    - Optional parameter.
    - 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.

4. dump_xml:
    - Optional parameter.
    - Outputs the xml requests & responses involved.

4.5 Compare_imc_managedobject

This operation compares managed objects.

```
compare_imc_managedobject(reference_object, difference_object,
exclude_different=YesOrNo.FALSE, include_equal=YesOrNo.FALSE,
no_version_filter=YesOrNo.FALSE, include_operational=YesOrNo.FALSE,
xlate_map=None)
```

Mandatory Input set:

(1) reference_object Objects used as a reference for comparison.

(2) difference_object Objects that are compared to the reference objects.

Parameter description:

1. reference_object:
   - Mandatory parameter.
   - Objects used as a reference for comparison.

2. difference_object:
   - Mandatory parameter.
   - Objects that are compared to the reference objects.

3. `exclude_different`:
   - Optional parameter.
   - Displays only the properties of compared objects that are equal.

4. `include_equal`:
   - Optional parameter.
   - Displays properties of compared objects that are equal.
     By default, only properties that differ between the reference and difference objects are displayed.

5. `no_version_filter`:
   - Optional parameter.
   - Ignore minimum version in properties.

6. `include_operational`:
   - Optional parameter.
   - Specified to include operational properties.

7. `xlate_map`:
   - Optional parameter.
   - Translation map with DNs of entities that needs to be translated.

4.6 sync_imc_managedobject

This operation syncs Managed Object of type ManagedObject. This operation takes the difference object (output of CompareManagedObject) and applies the differences on reference Managed Object.

```
sync_imc_managedobject(difference, delete_not_present=False,
no_version_filter=False, dump_xml=None)
```

Mandatory Input set:

(1) `difference` containing difference object to be synced

Parameter description:

1. `difference`:
   - Mandatory parameter.
   - Specifies the Difference object (output of compare_imc_managedobject) which has differences of the properties of two or more Managed Objects.

2. `delete_not_present`:
   - Mandatory parameter.
   - If specified, any missing MOs in reference Managed Object set will be deleted.

3. `no_version_filter`:
   - Optional parameter.
   - If specified, minversion for Mos or properties to be added in reference Managed Object will not be checked.

4. `dump_xml`:
   - Optional parameter.
   - Outputs the xml requests & responses involved.

4.7 get_imc_techsupport

This operation will create and download the technical support data for the respective IMC server.

get_imc_techsupport(remotehost, remotefile, protocol, username, password, timeout_sec, dump_xml=None)

Parameter description:

1. remotehost:
    - Mandatory parameter (string).
    - Remote host where username needs to download technical support data.
2. remotefile:
    - Mandatory parameter (string).
    - Filename with full path where username needs to download the technical support data file.

      Note: Should be a tar.gz file.
3. protocol:
    - Mandatory parameter (string).
    - Protocol used for transferring the file to remote host.
4. username:
    - Mandatory parameter (string).
    - Username to login to remote host.
5. password:
    - Mandatory parameter (string).
    - Password to login to remote host.
6. timeout_sec:
    - Optional parameter (string).
    - Waits for specified maximum time in sec for the tech support file to generate else exit.
7. dump_xml:
    - Optional parameter.
    - Outputs the xml requests & responses involved.

4.8 backup_imc

This operation will create and download the backup of IMC.

backup_imc(remotehost, remotefile, protocol, username, password, passphrase, timeout_sec, dump_xml=None)

Parameter description:

1. remotehost:
    - Mandatory parameter (string).
    - Remote host where username needs to download the Imc configuration.
2. remotefile:
    - Mandatory parameter (string).
    - Filename with full path where username needs to download the Imc configuration file.

      Note: Should be .xml file.
3. protocol:
    - Mandatory parameter (string).
    - Protocol used for transferring the file to remote host.
4. username:
    - Mandatory parameter (string).
    - Username to login to remote host.
5. password:
    - Mandatory parameter (string).
    - Password to login to remote host.
6. passphare:
    - Optional parameter (string).
    - Key for opening backup file.
7. timeout_sec:
    - Optional parameter (string).
    - Waits for specified maximum time in sec for the Imc configuration file to generate else exit.
8. dump_xml:
    - Optional parameter.
    - Outputs the xml requests & responses involved.

4.9 import_imc_backup

This operation will upload the IMC backup taken earlier via GUI or backup_imc operation for all configuration, system configuration, and logical configuration files. You can perform an import while the system is up and running.

import_imc_backup(remotehost, remotefile, protocol, username, password, passphrase, dump_xml=None)

Parameter description:

1. remotehost:
    - Mandatory parameter (string).
    - Remote host from where username needs to import the Imc configuration.
2. remotefile:
    - Mandatory parameter (string).
    - Filename with full path from where username needs to import the Imc configuration file.

      Note: Should be .xml file.
3. protocol:
    - Mandatory parameter (string).
    - Protocol used for transferring the file to remote host.
4. username:
    - Mandatory parameter (string).
    - Username to login to remote host.
5. password:
    - Mandatory parameter (string).
    - Password to login to remote host.
6. passphare:
    - Optional parameter (string).
    - Key for opening backup file.
7. dump_xml:
    - Optional parameter.
    - Outputs the xml requests & responses involved.

4.10 start_imc_iod_snapshot

This operation will start the IOD Snapshot.

start_imc_iod_snapshot(iso_share_ip, iso_share, iso_share_type, username, password, timeout, remote_share_ip, remote_share_path, remote_share_file, remote_share_type, remote_share_username, remote_share_password, admin_state="trigger",dump_xml=None)

Parameter description:

1. iso_share_ip:
    - Mandatory parameter (string).
    - IP Address of the host where the iso image resides.
2. iso_share:
    - Mandatory parameter (string).
    - Path of the iso image on the iso_share_ip.
3. iso_share_type:
    - Mandatory parameter (string).
    - This specifies the ISO share type.
4. username:
    - Mandatory parameter (string).
    - Username to login to iso_share_ip server.
5. password:
    - Mandatory parameter (string).
    - Password to login to iso_share_ip server.
6. timeout:
    - Optional parameter (string).
    - This specifies the timeout in minutes to exit the operation.
7. remote_share_ip:
    - Mandatory parameter (string).
    - IP Address of the remote server where the snapshot file will generate.
8. remote_share_path:
    - Mandatory parameter (string).
    - Path of the snapshot file.
9. remote_share_file:
    - Mandatory parameter (string).
    - Snapshot file name.
10. remote_share_type:
    - Mandatory parameter (string).
      This specifies the remote share type.
11. remote_share_username:
    - Mandatory parameter (string).
    - Username to login to remote_share_ip server.
12. remote_share_password:
    - Mandatory parameter (string).
    - Password to login to remote_share_ip server.

13. admin_state:
    - Mandatory parameter (string).
    - Specifies the admin state.
14. dump_xml:
    - Optional parameter.
    - Outputs the xml requests & responses involved.

4.11 stop_imc_iod_snapshot

This operation will start the IOD Snapshot.

stop_imc_iod_snapshot(timeout, admin_state="trigger",dump_xml=None)

Parameter description:

1. timeout:
    - Mandatory parameter (string).
    - This specifies the timeout in minutes to exit the operation.
2. admin_state:
    - Optional parameter (string).
    - Specifies the admin state. By default it is "trigger"
3. dump_xml:
    - Optional parameter.
    - Outputs the xml requests & responses involved.

4.12 update_imc_firmware

This operation will upload the firmware image for the respective managed object.

update_imc_firmware(in_mo, admin_state, protocol, share_type, remote_server, remote_path, username, password, secure_boot=None, dump_xml=None)

Parameter description:

1. in_mo:
    - Mandatory parameter (string).
    - Managed Object for which firmware image to be updated.
2. admin_state:
    - Mandatory parameter (string).
    - Specifies the admin state.
3. protocol:
    - Mandatory parameter (string).
    - Specifies the protocol for transferring the file to remote host.
4. share_type:
    - Mandatory parameter (string).
    - Specifies the share type.
5. remote_server:
    - Mandatory parameter (string).
    - IP Address of the remote server where the firmware image resides.
6. remote_path:
    - Mandatory parameter (string).
    - Path of the firmware image on the remote server.
7. username:
    - Mandatory parameter (string).
    - Username to login to remote server.
8. password:
    - Mandatory parameter (string).
    - Password to login to remote server.
9. secure_boot:
    - Optional parameter.
10. dump_xml:
    - Optional parameter.
    - Outputs the xml requests & responses involved.

4.13 update_imc_firmware_huu

This operation uploads the HUU firmware image for the respective managed object.

update_imc_firmware_huu(remote_share, share_type, remote_ip, username, password, update_component, stop_on_error=None, timeout=None, verify_update=None, cimc_secure_boot=None, dump_xml=None)

Parameter description:

1. remote_share:
    - Mandatory parameter (string).
    - Path of the firmware image on the remote Ip.
2. share_type:
    - Mandatory parameter (string).
    - Specifies the share type.
3. remote_ip:
    - Mandatory parameter (string).
    - IP Address of the remote server where the firmware image resides.
4. username:
    - Mandatory parameter (string).
    - Username credential to login to remote server.
5. password:
    - Mandatory parameter (string).
    - Password credential to login to remote server.
6. update_component:
    - Mandatory parameter (string).
    - This specify the component to update or “all”
7. stop_on_error:
    - Optional parameter (string).
    - This specifies the action or error. “yes” or “no”
8. timeout:
    - Optional parameter (string).
    - This specifies the timeout in minutes to exit the operation.
9. verify_update:
    - Optional parameter (string).
    - This specifies if Imc verify after update. “yes” or “no”
10. cimc_secure_boot:
    - Optional parameter.
11. dump_xml:
    - Optional parameter.
    - Outputs the xml requests & responses involved.

4.14 export_imc_session

This operation will store the credential of currently logged in IMC in current session to a file. Password will be stored in encrypted format using key.

export_imc_session(file_path, key, merge=YesOrNo.FALSE)

Mandatory Input sets:

(1) export_imc_session(file_path, key, merge=True<Optional>)

Parameter description:

1. file_path:
    - Mandatory parameter (string).
    - Path of the .xml credential file.
2. key:
    - Mandatory parameter (string).
    - String used for encrypting password in .xml credential file.
3. merge:
    - Optional parameter (boolean).
    - It should be **False** unless username wants to append the existing credential file with new credential.

4.15 import_imc_session

This operation will do a login to each IMC which is present in credential file.

import_imc_session(file_path, key)

Mandatory Input sets:

(1) import_imc_session(file_path, key)

Parameter description:

1. file_path:
    - Mandatory parameter (string).
    - Path of the credential file.
2. key:
    - Mandatory parameter (string).
    - String used while export_imc_session operation.

4.16 Methods

4.16.1 aaa_get_compute_auth_tokens

```
aaa_get_compute_auth_tokens(dump_xml=None)
```

4.16.2 aaa_keep_alive

```
aaa_keep_alive(dump_xml=None)
```

4.16.3 aaa_login

```
aaa_login(inName, inPassword, dump_xml=None)
```

4.16.4 aaa_logout

```
aaa_logout(dump_xml=None)
```

4.16.5 aaa_refresh

```
aaa_refresh(inName, inPassword, dump_xml=None)
```

4.16.6 config_conf_mo

```
config_conf_mo(dn, in_config, in_hierarchical=YesOrNo.FALSE, dump_xml=None)
```

4.16.7 config_resolve_children

The configResolveChildren method retrieves children of managed objects under a specific DN in the managed information tree. A filter can be used to reduce the number of children being returned.

```
config_resolve_children(class_id, inDn, inFilter,
                                in_hierarchical=YesOrNo.FALSE, dump_xml=None)
```

4.16.8 config_resolve_class

The configResolveClass method returns requested managed object in a given class. If in_hierarchical=true,then the results contain children.

```
config_resolve_class(class_id, inFilter, in_hierarchical=YesOrNo.FALSE,
dump_xml=None)
```

4.16.9 config_resolve_dn

For a specified DN, the configResolveDn method retrieves a single managed object.

```
config_resolve_dn(dn, in_hierarchical=YesOrNo.FALSE, dump_xml=None)
```

4.16.10 config_resolve_parent

For a specified DN, the configResolveParent method retrieves the parent of the managed object.

```
config_resolve_parent(dn, in_hierarchical=YesOrNo.FALSE, dump_xml=None)
```

4.16.11 event_subscribe

```
event_subscribe(dump_xml=None)
```

4.16.12 event_unsubscribe

```
event_unsubscribe(dump_xml=None)
```

5 Samples

A couple of samples are packaged with the SDK Tar.

References

The following documents were referred to while creating this material:

1. Cisco IMC Overview.
   https://developer.cisco.com/site/data-center/ucs-system-mgmt/ucs-dev-center/overview/ucs-c-series/

2. Cisco IMC Manager CLI Configuration Guide
   http://www.cisco.com/en/US/docs/unified_computing/ucs/c/sw/cli/config/guide/151/b_Cisco_UCS_C-series_CLI_Configuration_Guide_151.html