Cisco UCS Director REST API Cookbook

Release 1.0
Published: April, 2015

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide.
Addresses, phone numbers, and fax numbers are listed on the Cisco website at:
www.cisco.com/go/offices.

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING,WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.

© 2015 Cisco Systems, Inc. All rights reserved.

Table of Contents

1 Getting Started....4
2 REST Tools....4
2.1 REST API Browser....4
2.1.1 Usage Example 1: Retrieving the VM Summary....5
2.1.2 Usage Example 2: Creating a Group....6
2.1.3 Sample Java Code....7
3 API Usage Examples....9
3.1 Administration....9
3.1.1 Creating a Group....9
3.1.2 Listing the Groups....11
3.1.3 Modifying a Group....11
3.1.4 Deleting a Group....11
3.2 Workflow Orchestration....12
3.2.1 Submitting a Service Request....12
3.2.2 Submitting a VApp Service Request....12
3.3 VM Management....12
3.3.1 Powering On a VM....12
3.3.2 Powering Off a VM....13
3.3.3 Rebooting a VM....13
3.3.4 Creating and resizing a VM Disk....13
3.3.5 Setting up a VMware VM Guest and Executing VIX Script 14
3.3.6 Deleting a VMware Snapshot....14
3.4 Managing Physical Accounts....15
3.4.1 Creating a Physical Account....15
3.4.2 Deleting a Physical Account....15
3.4.3 Listing the Accounts....15
3.5 Managing Catalogs....16
3.5.1 Creating a Catalog Item....16
3.5.2 Retrieving the Catalog Details....16
3.5.3 Deleting a Catalog Item....17
3.6 Managing VDCs....17
3.6.1 Listing the VDCs....17
3.6.2 Creating a VDC....17
3.6.3 Exporting a VDC....18
3.6.4 Importing a VDC....18
3.6.5 Deleting a VDC....18
3.7 Managing Containers....19
3.7.1 Retrieving a Service Container....19
3.7.2 Retrieving a Service Container with Catalog....19
3.7.3 Deleting a Service Container....19

1 Getting Started

The cookbook provides a gist of the REST services offered by Cisco UCS Director. The cookbook is expected to aid a Developer with, but not restricted to, the following:

- Demonstrate the REST-related tools available with Cisco UCS Director.
- Automate the orchestration of tasks through the REST APIs .

2 REST Tools

By default, Cisco UCS Director is enabled to respond to REST API requests from the appropriate REST clients.

Cisco UCS Director provides the following entities to aid in REST development:

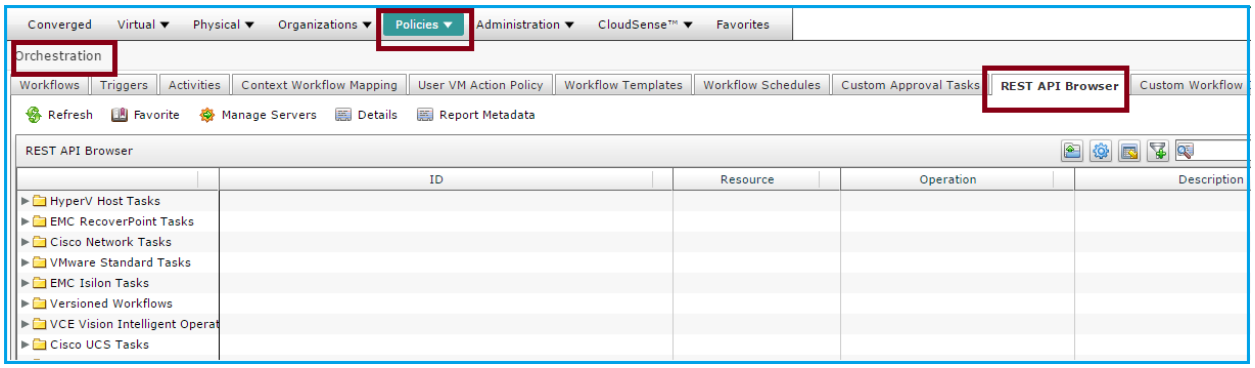
- REST API Browser
- REST API SDK Bundle

For more information on the REST API SDK bundle, see the Cisco UCS Director REST API Developer Guide.

2.1 REST API Browser

The REST API browser available with Cisco UCS Director aids the Developers and QA Engineers to validate the REST APIs added to the framework.

An administrator or group administration user can launch the REST API Browser. To launch the REST API Browser, choose **Policies > Orchestration > REST API Browser**.



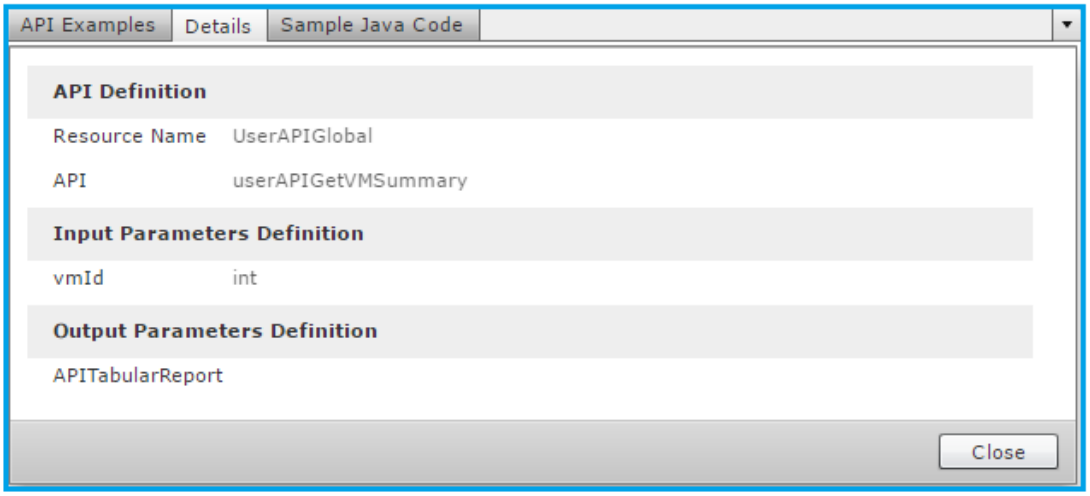
The REST API browser groups the APIs based on the tasks needed to manage the respective infrastructure components of the datacenter.

To view an API, expand a task folder and double-click the report for the respective API. The browser provides the following tabs for the selected API:

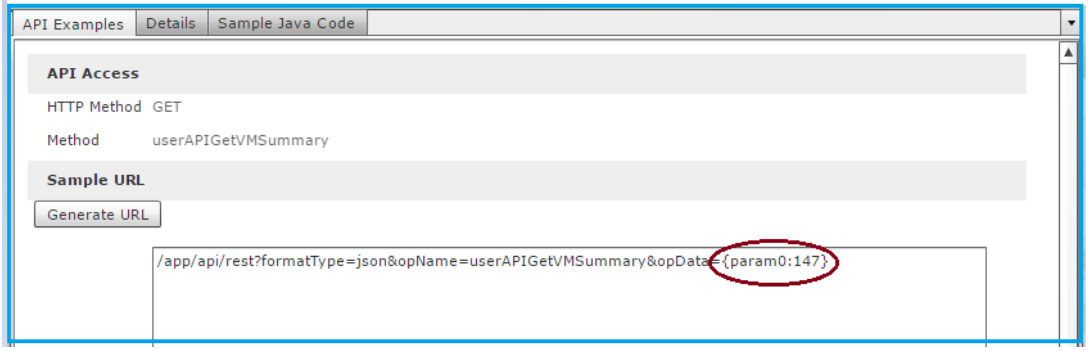
1) API Examples—Displays the inputs for user-selection. A Developer can provide his inputs for the respective API as required and generate a sample URL.
2) Details—Defines the syntax and the semantics for the API.
3) Sample Java Code—A Java code snippet that demonstrates the use of the API.

2.1.1 Usage Example 1: Retrieving the VM Summary

Click the **Details** tab, to view the definition of the userAPIGetVMSummary API. The userAPIGetVMSummary API takes the **vmId** as its input.



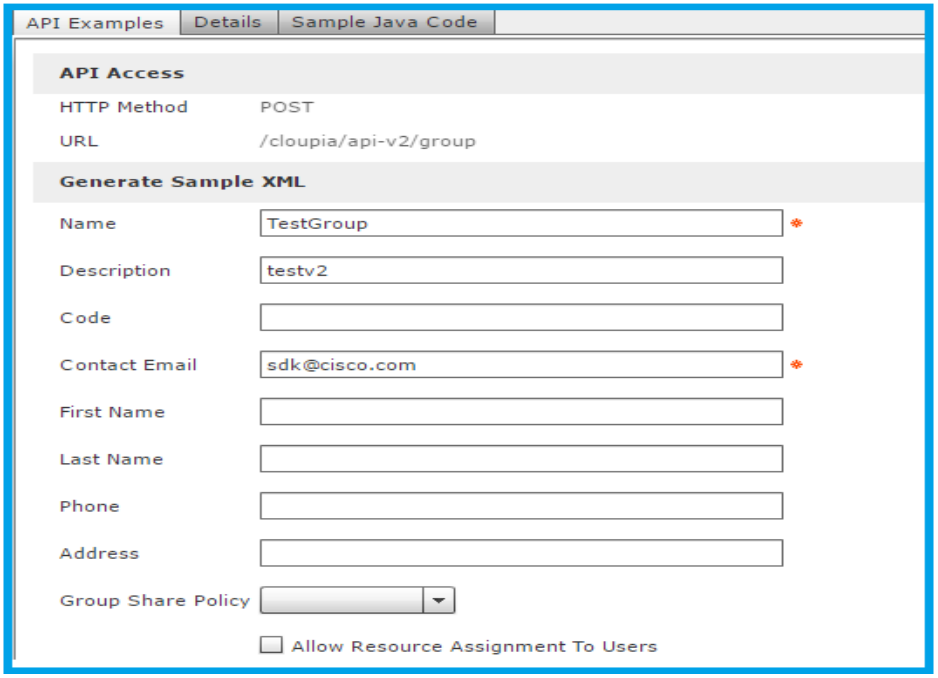
In the **API Examples** tab, click **Generate URL** to get the URL for the REST operation. The **vmId** must be passed as the param0, as shown in the following dialog box.



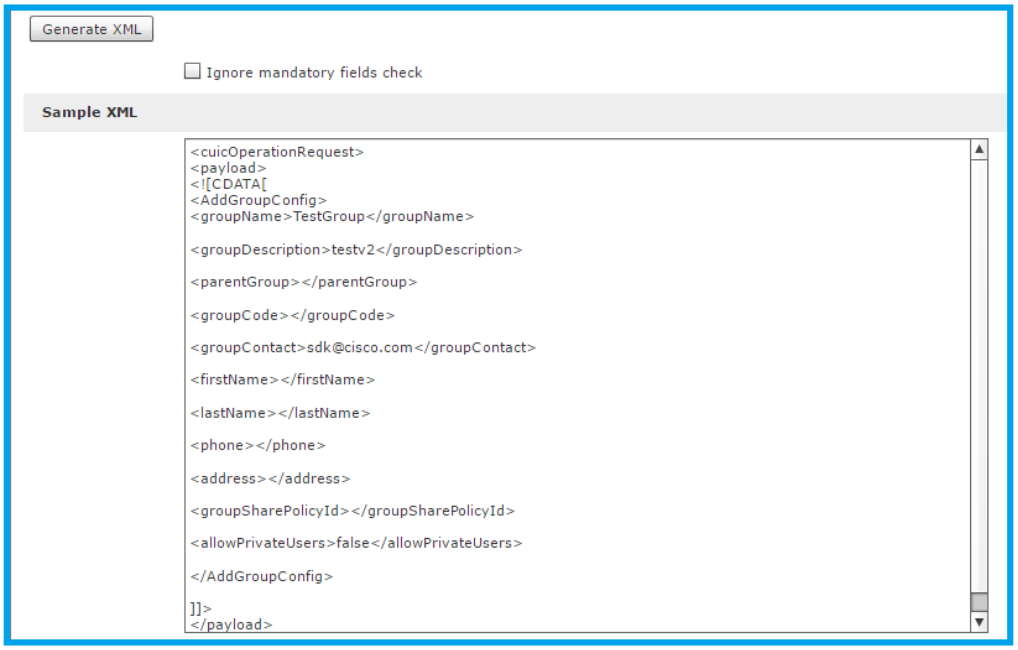
From the **UCSD Server** drop-down list, choose Cisco UCS Director on which the API is targeted at and click **Execute REST API** for a response.

2.1.2 Usage Example 2: Creating a Group

The following UI is presented to create a group.



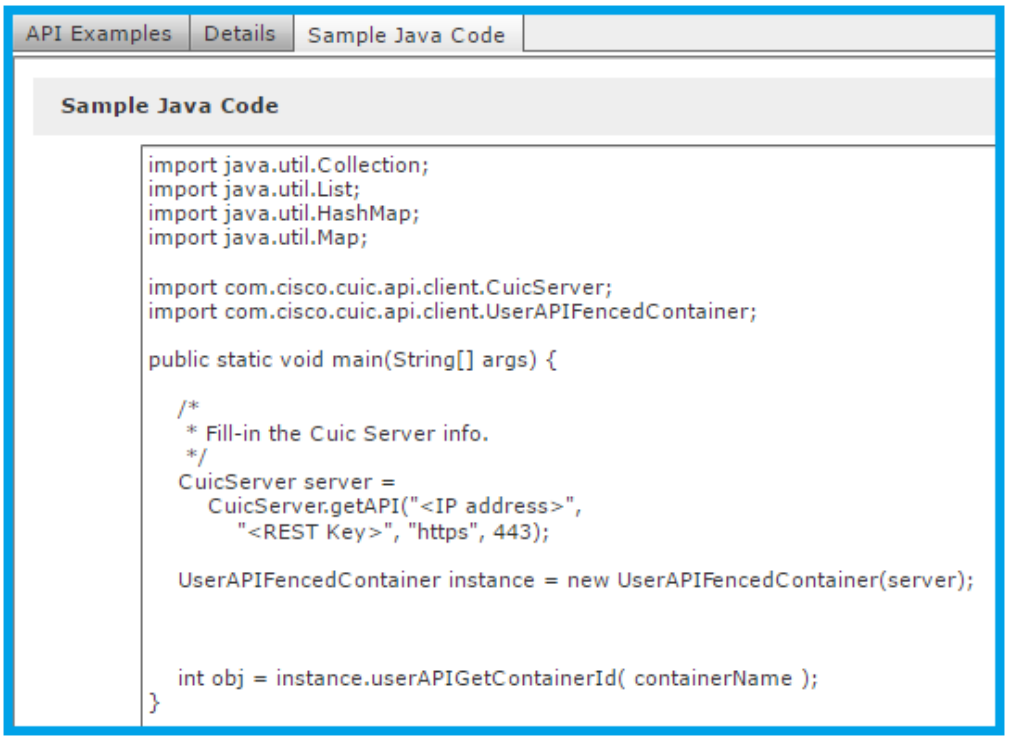
Fill in the fields and click **Generate XML** to generate the sample XML file for group creation.



From the **UCSD Server** drop-down list, choose Cisco UCS Director on which the API is targeted at and click **Execute REST API** for a response.

2.1.3 Sample Java Code

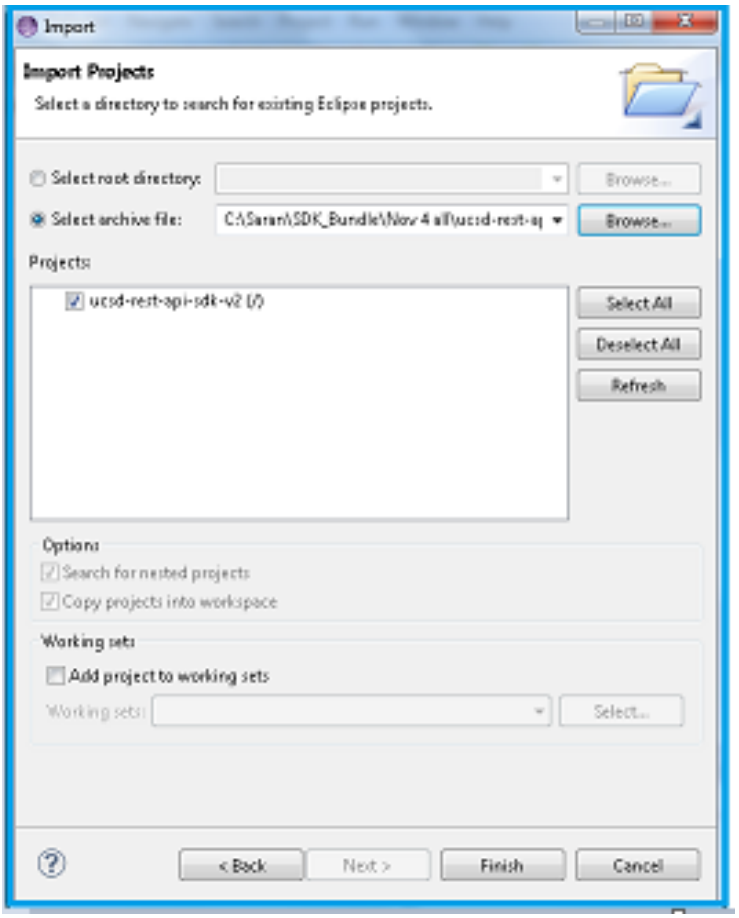
The **Sample Java Code** tab provides the code snippet that can be used to automate the management services through code.



The code snippet can be used to execute the respective API. To execute the code in the Eclipse and obtain the output, you must import the SDK bundle as a Java Project into the Eclipse IDE.

To import the SDK bundle, do the following:

1. In the Eclipse IDE, choose **File > New > Java Project**.
   The **Create a Java Project** dialog box appears.
2. In the **Project Name** field, enter a name for the project.
3. Right click the project, and select **Import**.
4. In the **Import** dialog box, select **Existing projects into Workspace** and click **Next**.
5. Click **Browse** and navigate to the folder where you extracted the SDK Bundle.



6. Click **Finish**.

The Eclipse IDE displays the SDK Bundle project on the **Project Explorer** tab.

3 API Usage Examples

This section captures code examples on the API usage. For detailed explanation on the respective APIs, refer to the Cisco UCS Director REST API Developer Guide.

3.1 Administration

3.1.1 Creating a Group

There are a couple of APIs to create a group, demonstrated as follows:

```
public class userAPICreateGroup {
public static void main(String[] args) throws Exception {
    CuicServer server =
    CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369F0
    FC64", "https", 443);

    UserAPIGlobal instance = new UserAPIGlobal(server);
    APIGroup apiGroup = new APIGroup();
    apiGroup.setGroupId(1);
    apiGroup.setGroupName("TestGrp");
    apiGroup.setDescription("Testing sample");
    apiGroup.setParentGroupId(0);
    apiGroup.setParentGroupName("ABC");
    apiGroup.setEmailAddress("sdk@example.com");
    apiGroup.setLastName("l");
    apiGroup.setFirstName("f");
    apiGroup.setPhoneNumber("12345");
    apiGroup.setAddress("xyz");
    apiGroup.setGroupType(0);
    int obj = instance.userAPICreateGroup( apiGroup );
    System.out.println(obj);
      }
}
```

The following approach and usage of another API is equivalent to executing the workflow task that creates a group.

```
public class userAPIGroupcreate {
public static void main(String[] args) throws Exception {
       CuicServer server =
       CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369
       F0FC64", "https", 443);

       AddGroupConfig instance = new AddGroupConfig(server);
       instance.setGroupName("testgroup");
       instance.setGroupDescription("testing");
       instance.setParentGroup("testparentgroup");
       instance.setGroupCode("1256");
       instance.setGroupContact("email@example.com");
       instance.setFirstName("abcd");
       instance.setLastName("dabc");
       instance.setPhone("1234");
       instance.setAddress("test");
       instance.setGroupSharePolicyId("abcd123");
       instance.setAllowPrivateUsers(false);
       AddGroupConfigResponse obj = instance.execute();
        }
}
```

Cisco UCS Director provides API wrappers for execution of individual workflow tasks. There are cases where more than one APIs that does the same job are

available with the SDK bundle, as seen in this section. Thus, a Developer shall use either of those APIs as explained above.

3.1.2 Listing the Groups

```
public class userAPIGetGroups {
public static void main(String[] args) throws Exception {
    CuicServer server =
    CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369F0
    FC64", "https", 443);

   UserAPIGlobal instance = new UserAPIGlobal(server);
    APIGroup grp = new APIGroup();
    System.out.println("Group Id :" +grp.getGroupId());
    System.out.println("Group Name :" +grp.getGroupName());
    System.out.println("Description :" +grp.getDescription());
    System.out.println("Parent Group ID :"
    +grp.getParentGroupId());
    System.out.println("Parent Group Name :"
    +grp.getParentGroupName());
    System.out.println("Email Address :" +grp.getEmailAddress());
    System.out.println("Last Name :" +grp.getLastName());
    System.out.println("First Name :" +grp.getFirstName());
    System.out.println("Phone number :" +grp.getPhoneNumber());
    System.out.println("Address :" + grp.getAddress());
    System.out.println("Group Type : " + grp.getGroupType());
    List<APIGroup> obj = instance.userAPIGetGroups();
   System.out.println(grp);
       }
}
```

3.1.3 Modifying a Group

```
public class userAPIUpdateGroup {
public static void main(String[] args) throws Exception {
    CuicServer server =
    CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369F0
    FC64", "https", 443);

    UserAPIGlobal instance = new UserAPIGlobal(server);
    APIGroup apiGroup = new APIGroup();
    apiGroup.setGroupName("G1");
    apiGroup.setEmailAddress("123@example.com");
    apiGroup.setDescription("update test");
    boolean obj = instance.userAPIUpdateGroup(apiGroup);
    System.out.println(obj);
       }
}
```

3.1.4 Deleting a Group

```
public class userAPIDeleteGroup {
public static void main(String[] args) throws Exception {
```

```
        CuicServer server =
        CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369
        F0FC64", "https", 443);

        UserAPIGlobal instance = new UserAPIGlobal(server);
        boolean obj = instance.userAPIDeleteGroup( 10 );
        System.out.println(obj);
        }
}
```

3.2 Workflow Orchestration

3.2.1 Submitting a Service Request

```
public class userAPISubmitServiceRequest{
public static void main(String[] args) throws Exception {
        CuicServer server =
        CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369
        F0FC64", "https", 443);

        UserAPIGlobal instance = new UserAPIGlobal(server);
        int obj = instance.userAPISubmitServiceRequest( "vmware1" ,
        "vmware", 1, 1, 1, "test");
        System.out.println(obj);
        }
}
```

3.2.2 Submitting a VApp Service Request

```
public class userAPISubmitVAppServiceRequest {
public static void main(String[] args) throws Exception {
   CuicServer server =
   CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369F0F
   C64", "https", 443);

    UserAPIGlobal instance = new UserAPIGlobal(server);
    APINameValueList list = new APINameValueList();
    APINameValue nv = new APINameValue();
    nv.setName("Name");
    nv.setValue("value");
        list.addNameValue(nv );
        int obj = instance.userAPISubmitVAppServiceRequest( "CatAD"
, list );
        }
}
```

3.3 VM Management

3.3.1 Powering On a VM

```
public class userAPIExecuteVMAction {
public static void main(String[] args) throws Exception {
     CuicServer server =
     CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369F
     0FC64", "https", 443);
```

```
        UserAPIGlobal instance = new UserAPIGlobal(server);
        String obj = instance.userAPIExecuteVMAction( 1 , "powerOn"
        , "Testing" );
         }
}
```

3.3.2 Powering Off a VM

```
public class userAPIExecuteVMAction {
public static void main(String[] args) throws Exception {

    CuicServer server =
    CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369F0
    FC64", "https", 443);

    UserAPIGlobal instance = new UserAPIGlobal(server);
    String obj = instance.userAPIExecuteVMAction( 1 , "powerOff"
    , "Testing" );
   }
}
```

3.3.3 Rebooting a VM

```
public class userAPIExecuteVMAction {
public static void main(String[] args) throws Exception {
    CuicServer server =
    CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369F0
    FC64", "https", 443);

    UserAPIGlobal instance = new UserAPIGlobal(server);
    String obj = instance.userAPIExecuteVMAction( 1 , "Reboot" ,
    "Testing" );
      }
}
```

3.3.4 Creating and resizing a VM Disk

```
public class VMDiskcreateandresize{
public static void main(String[] args) throws Exception {
    CuicServer server =
    CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369F0
    FC64", "https", 443);

    CreateVMDisk instance = new CreateVMDisk(server);
    instance.setVmId(120);
    instance.setDiskSize("20");
    instance.setFromStoragePolicy(false);
    instance.setDiskType("Database");
    instance.setDatastoreName("Test123");
    instance.setThinProvision(false);
    instance.execute();

    System.out.println(""+instance.getDiskSize());
    System.out.println(""+instance.getVmId());
    System.out.println(""+instance.getDiskType());
```

```
ResizeVMDisk instanceres = new ResizeVMDisk(server);
instanceres.setVmId(instance.getVmId());
instanceres.setVmDisk("tempappvm1");
instanceres.setProvisionedDisk(instance.getDiskType());
instanceres.setSize("40");
instanceres.execute();}}
```

3.3.5 Setting up a VMware VM Guest and Executing VIX Script

```
public class vmwareguestandvix{
public static void main(String[] args) throws Exception {
    CuicServer server =
    CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369F0
    FC64", "https", 443);

    GuestSetup instance = new GuestSetup(server);
    instance.setVmId(120);
    instance.setCredentialsOptions("Do not Share");
    instance.setUserId("admin");
    instance.setPassword("admin");
    GuestSetupResponse obj = instance.execute();

    System.out.println(""+instance.getUserId());
    System.out.println(""+instance.getVmId());
    System.out.println(""+instance.getPassword());

    ExecuteVIXScript instancevix = new ExecuteVIXScript(server);
    instancevix.setAccountName("cloud123");
    instancevix.setVmId(instance.getVmId());
    instancevix.setCredentialType("Login");
    instancevix.setLogin(instance.getUserId());
    instancevix.setPassword(instance.getPassword());
    instancevix.setScript("/bin/date");
    instancevix.setUndoScript("");
    instancevix.setUndoScriptTask(false);
    instancevix.setOutputDisplay(false);
    ExecuteVIXScriptResponse obj = instancevix.execute();}}
```

3.3.6 Deleting a VMware Snapshot

```
public class DeleteVMsnap{
public static void main(String[] args) throws Exception {
       CuicServer server =
       CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369
       F0FC64", "https", 443);

        DeleteVMSnapshot instance = new DeleteVMSnapshot(server);
       instance.setVmId(168);
       instance.setSnapshotName("test");
       instance.setDeleteChild(false);
        DeleteVMSnapshotResponse obj = instance.execute();
        }
```

```
}
```

3.4 Managing Physical Accounts

3.4.1 Creating a Physical Account

```
public class userAPIcreateInfraAccount {
public static void main(String[] args) throws Exception {
CuicServer server =
CuicServer.getAPI("192.0.2.187","6C6416AD90704DF495A6B4D0A75A0BB1
", "https", 443);

UserAPIAccounts instance = new UserAPIAccounts(server);

//Example of UCSM Account

InfraAccountDetails infraAccountDetails = new
InfraAccountDetails();
    infraAccountDetails.setAccountName("AccName");
    infraAccountDetails.setPodName("PodName");
    infraAccountDetails.setAccountCategory(1);
    infraAccountDetails.setAccountType("11");
    infraAccountDetails.setProtocol("http");
    infraAccountDetails.setPort(80);
    infraAccountDetails.setDestinationIPAddress("1.2.3.4");
    infraAccountDetails.setLogin("abc");
    infraAccountDetails.setPassword("pswd");
boolean obj =
instance.userAPICreateInfraAccount(infraAccountDetails );
System.out.println(obj);
      }
}
```

3.4.2 Deleting a Physical Account

```
public class userAPIDeleteInfraAccount {
public static void main(String[] args) throws Exception {
    CuicServer server =
    CuicServer.getAPI("192.0.2.207","1A8DE698E2BF4C0B989476A369F0
    FC64", "https", 443);

    UserAPIAccounts instance = new UserAPIAccounts(server);
    boolean obj = instance.userAPIDeleteInfraAccount(
    "AccountName" );
    System.out.println("Is the Account deleted ? :" +obj);
      }
}
```

3.4.3 Listing the Accounts

```
public class userAPIGetAllAccounts {
public static void main(String[] args) throws Exception {
      CuicServer server =
```

```
        CuicServer.getAPI("192.0.2.207",
        "1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);

        UserAPIAccounts instance = new UserAPIAccounts(server);
        List<String> obj = instance.userAPIGetAllAccounts();
        String[] strarray = new String[obj.size()];
        obj.toArray(strarray);
        String[] tmp = new String[strarray.length];
        for (int i = 0; i < tmp.length; ++i) {
              tmp[i] = (String) strarray[i];
              System.out.println(tmp[i]);
}
}
}
```

3.5 Managing Catalogs

3.5.1 Creating a Catalog Item

```
public class  CreateCatalogItem {
public static void main(String[] args) throws Exception {
CuicServer server =
CuicServer.getAPI("192.0.2.207",
"1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);

UserAPIGlobal instance = new UserAPIGlobal(server);

APICatalogItem item = new APICatalogItem();
    item.setCatalogItemId(20);
    item.setCatalogItemName("sample");
    item.setCatalogItemDescription("test");
    item.setCloudName("sample");
    item.setImageId("sample");
    item.setGroups("sample");
    item.setAppliedToAllGroups(true);
    item.setSupportEmail("sample");
    item.setVdcCategoryId(1000);
    int[] appList = null ;
    appList[0]=1;
    appList[1]=1;
    item.setAppList(appList);
      item.setOtherApps("sample");
      item.setOtherOS("sample");
boolean b = userAPICreateCatalogItem(item);
      }
}
```

3.5.2 Retrieving the Catalog Details

```
public class userAPIgetCatalogDetails {
public static void main(String[] args) throws Exception {
      CuicServer server =
      CuicServer.getAPI("192.0.2.207",
      "1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);
```

```
        UserAPIGlobal instance = new UserAPIGlobal(server);
        APIProvisionParams obj = instance.userAPIGetCatalogDetails(
        "sample" );
        }
}
```

3.5.3 Deleting a Catalog Item

```
public class userAPIDeleteCatalogItem {
public static void main(String[] args) throws Exception {
        CuicServer server =
        CuicServer.getAPI("192.0.2.207",
        "1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);

        UserAPIGlobal instance = new UserAPIGlobal(server);
         boolean obj = instance.userAPIDeleteCatalogItem("sample");
        }
}
```

3.6 Managing VDCs

3.6.1 Listing the VDCs

```
public class userAPIgetallVDCs {
public static void main(String[] args) throws Exception {
        CuicServer server =
        CuicServer.getAPI("192.0.2.207",
        "1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);

        UserAPIGlobal instance = new UserAPIGlobal(server);
        APITabularReport obj = instance.userAPIGetAllVDCs();
        }
}
```

3.6.2 Creating a VDC

```
public class CreateVDC {
public static void main(String[] args) throws Exception {
    CuicServer server =
    CuicServer.getAPI("192.0.2.207",
    "1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);

    UserAPIGlobal instance = new UserAPIGlobal(server);
    APIVDCDetails vdcDetails = new APIVDCDetails() ;
    vdcDetails.setVdcName("sample");
    vdcDetails.setVdcDescription("sample");
    vdcDetails.setCloudName("sample");
    vdcDetails.setGroupName(1000);
    vdcDetails.setApprover1("sample");
    vdcDetails.setApprover2("sample");
    vdcDetails.setVdcSupportEmail("sample");
    vdcDetails.setVdcCustomerNoticationEmail("sample");
    vdcDetails.setSystemPolicy("sample");
```

```
    vdcDetails.setSlaPolicy("sample");
    vdcDetails.setComputingPolicy("sample");
    vdcDetails.setNetworkPolicy("sample");
    vdcDetails.setStoragePolicy("sample");
    vdcDetails.setCostModel("sample");
    vdcDetails.setLocked(true);
    vdcDetails.setDeletable(true);
    vdcDetails.setInactivityPeriodForDeletion(1000);
    boolean obj = instance.userAPICreateVDC( vdcDetails  );
      }
}
```

3.6.3 Exporting a VDC

```
public class ExportVDC {
public static void main(String[] args) throws Exception {
      CuicServer server =
      CuicServer.getAPI("192.0.2.207",
      "1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);

      UserAPIGlobal instance = new UserAPIGlobal(server);
      String obj = instance.userAPIExportVDC("sample" );
      }
}
```

3.6.4 Importing a VDC

```
public class importVDC {
public static void main(String[] args) throws Exception {
      CuicServer server =
      CuicServer.getAPI("192.0.2.207",
      "1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);

      UserAPIGlobal instance = new UserAPIGlobal(server);
      VDC obj = instance.userAPIImportVDC("sample" );
      }
}
```

3.6.5 Deleting a VDC

```
public class DeleteVMsnap{
public static void main(String[] args) throws Exception {
     CuicServer server =
     CuicServer.getAPI("192.0.2.207",
     "1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);
     CreateVdc instance = new CreateVdc(server);
     instance.setNetdevice("test");
     instance.setVdcName("test123");
     instance.setModuleType("example");
     instance.setHaPolicy(false);
     instance.setHaPolicyType("Hadefaultpolicy");
     instance.setHaPolicyAction("test1234");
     instance.setVdcNumber("123");
     instance.setPassword("sample");
     instance.setVdcTemplateName("VNIC template");
     instance.setVdcType("vmware");
```

```
        instance.setNetQosPolicy("QOSpolicy");
        instance.setCopyRunToStartConfig(false);
         CreateVdcResponse obj = instance.execute();
         }
}
```

3.7 Managing Containers

3.7.1 Retrieving a Service Container

```
public class GetServiceContainerData {
public static void main(String[] args) throws Exception {
     CuicServer server =
     CuicServer.getAPI("192.0.2.207",
     "1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);
      UserAPIFencedContainer instance = new
      UserAPIFencedContainer(server);
       ContainerDataObjects obj =
       instance.userAPIGetServiceContainerData(1000);
       }
}
```

3.7.2 Retrieving a Service Container with Catalog

```
public class CreateServiceContainer {
public static void main(String[] args) throws Exception {
       CuicServer server =
       CuicServer.getAPI("192.0.2.207",
        "1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);

       UserAPIAPICContainer instance = new
       UserAPIAPICContainer(server);
       int obj =
       instance.userAPICreateServiceContainerWithoutCatalog(
       "sample" , 1000 , "sample" );
       }
}
```

3.7.3 Deleting a Service Container

```
public class DeleteServiceContainerData {
public static void main(String[] args) throws Exception {
       CuicServer server =
       CuicServer.getAPI("192.0.2.207",
        "1A8DE698E2BF4C0B989476A369F0FC64", "https", 443);

       UserAPIFencedContainer instance = new
       UserAPIFencedContainer(server);
       int obj = instance.userAPIDeleteServiceContainer( 1000 );
       }
}
```