Cisco UCS Director REST API の手順書

リリース 1.0
2015 年 4 月に発行

Cisco Systems, Inc.
www.cisco.com

シスコは世界各国 200 箇所にオフィスを開設しています。
各オフィスの住所、電話番号、FAX 番号は次のシスコ Web サイトをご覧ください。
www.cisco.com/go/offices

目次

1 使用する前に

この手順書では、Cisco UCS Director が提供する REST サービスの要旨について説明しています。この手順書は、開発者が次のことを行うときに役立つことが期待されますが、それに限定されるものではありません。

- Cisco UCS Director で使用可能な REST 関連ツールのデモ。
- REST API を介したタスクのオーケストレーションの自動化。

2 REST ツール

デフォルトでは、Cisco UCS Director は適切な REST クライアントからの REST API 要求に応答できるようになっています。

Cisco UCS Director は、REST 開発に役立つ次のエンティティを提供します。

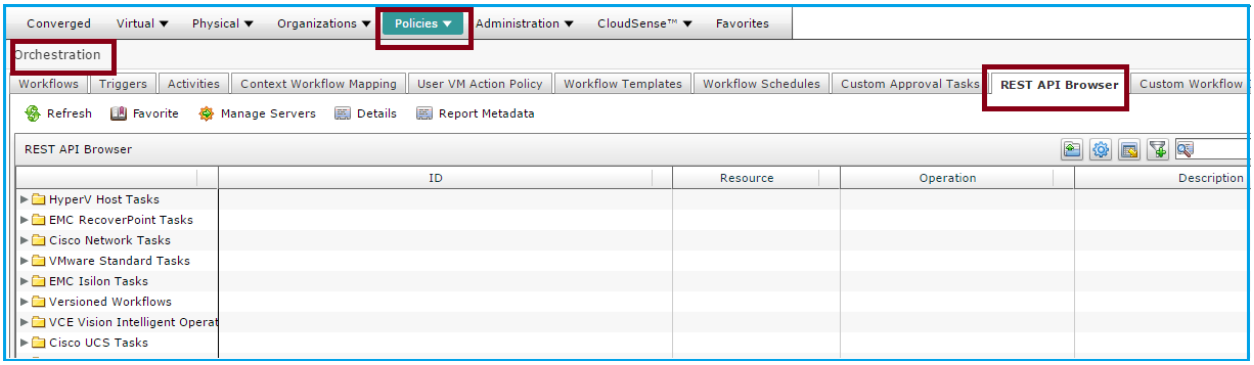
- REST API ブラウザ
- REST API SDK バンドル

REST API SDK バンドルの詳細については、『Cisco UCS Director REST API Developer Guide』を参照してください。

2.1 REST API ブラウザ

Cisco UCS Director で使用可能な REST API ブラウザは、開発者および QA エンジニアがフレームワークに追加された REST API を検証するのに役立ちます。

管理者またはグループ管理ユーザが REST API ブラウザを起動できます。REST API ブラウザを起動するには、[ポリシー（Policies）] > [オーケストレーション（Orchestration）] > [REST APIブラウザ（REST API Browser）] を選択します。



REST API ブラウザは、データセンターのそれぞれのインフラストラクチャ コンポーネントを管理するために必要なタスクに基づいて API をグループ化します。

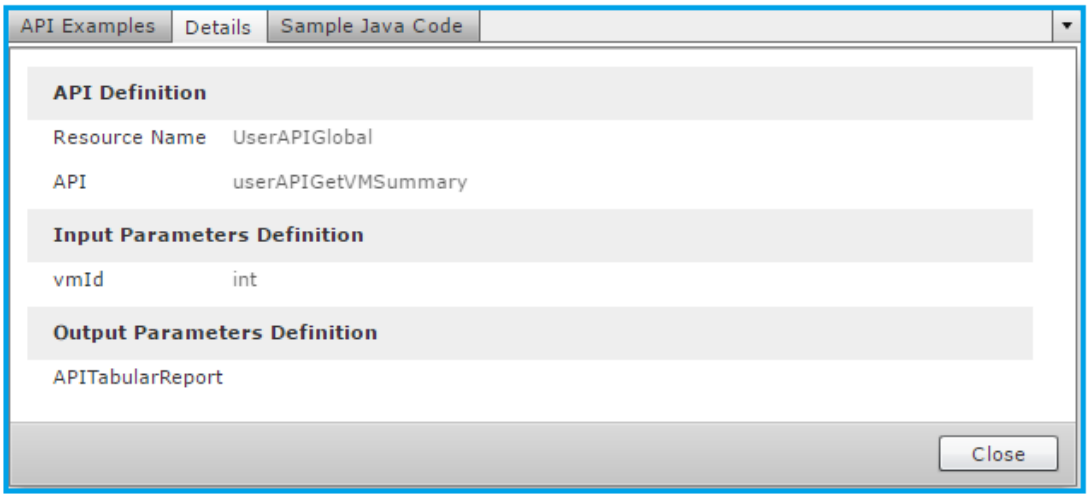
API を表示するには、タスク フォルダを展開し、それぞれの API のレポートをダブルクリックします。ブラウザでは、選択した API に関する次のタブが示されます。

1) [APIの例（API Examples）]：ユーザによる選択用の入力が表示されます。開発者は、必要に応じて各 API に独自の情報を入力し、サンプル URL を生成できます。
2) [詳細（Details）]：API のシンタックスおよびセマンティクスを定義します。
3) [サンプルJavaコード（Sample Java Code）]：API の使用を示す Java コード スニペット。

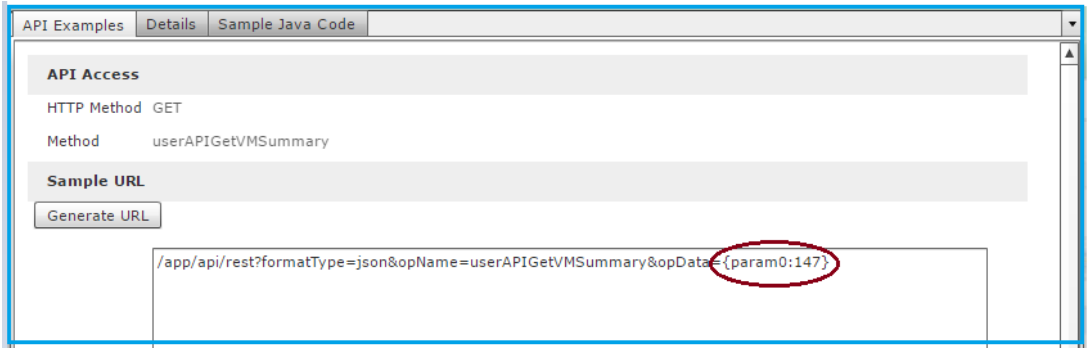
2.1.1 使用例 1：VM 概要の取得

userAPIGetVMSummary API の定義を表示するには、[詳細（Details）] タブをクリックします。

userAPIGetVMSummary API は、その入力として **vmId** を取得します。



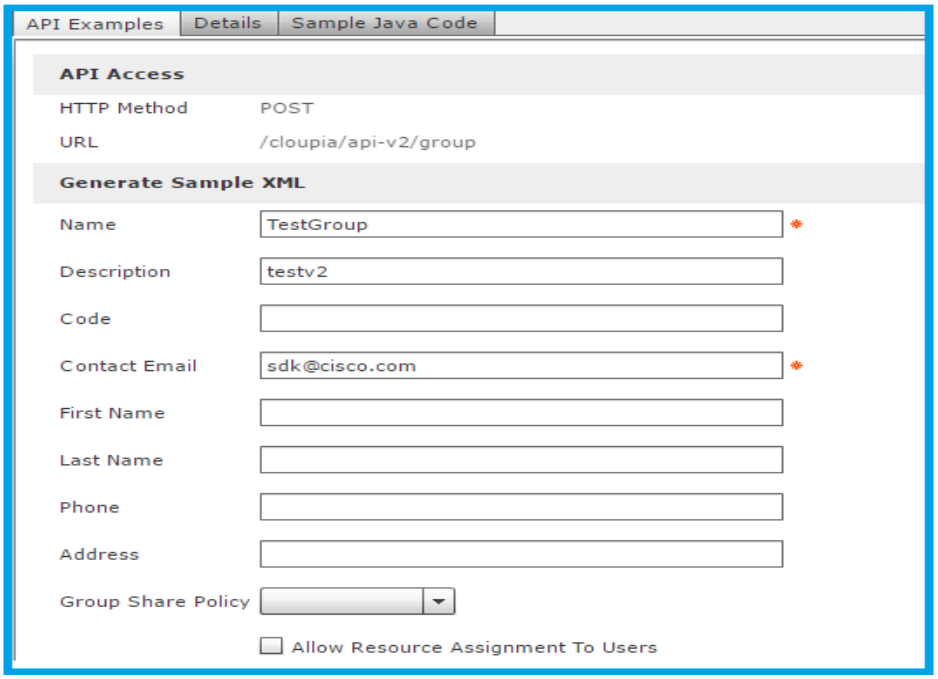
[APIの例（API Examples）] タブで、[URLの生成（Generate URL）] をクリックし、その REST 操作の URL を取得します。次のダイアログボックスに示すように、**vmId** は param0 として渡す必要があります。



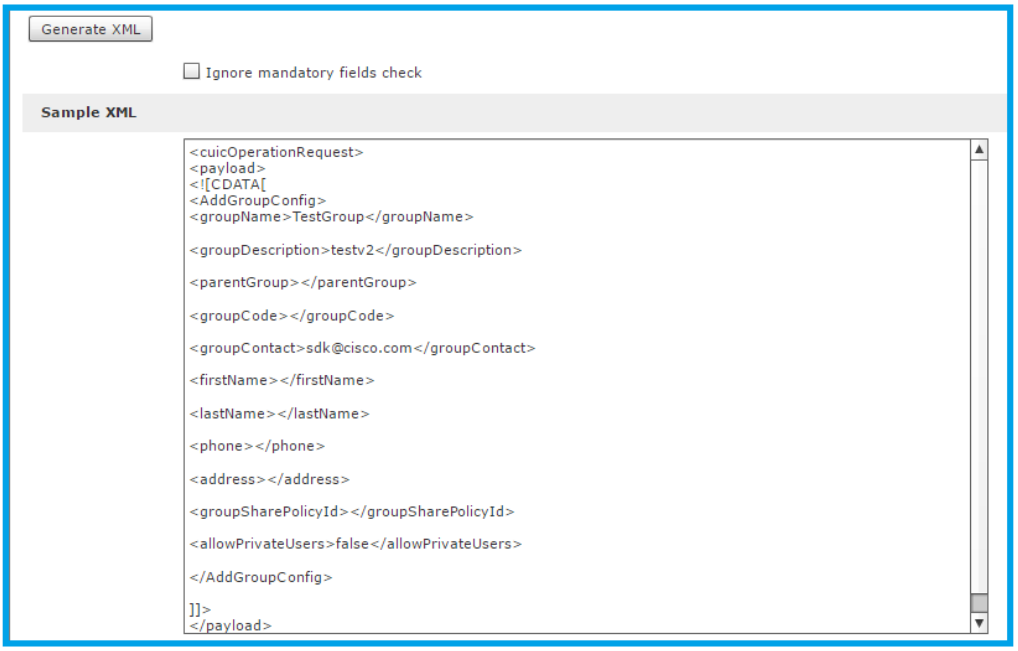
[UCSDサーバ（UCSD Server）] ドロップダウンリストから、API がターゲットとなる Cisco UCS Director を選択し、応答に対し [REST APIの実行（Execute REST API）] をクリックします。

2.1.2 使用例 2：グループの作成

次の UI は、グループを作成するために示されたものです。



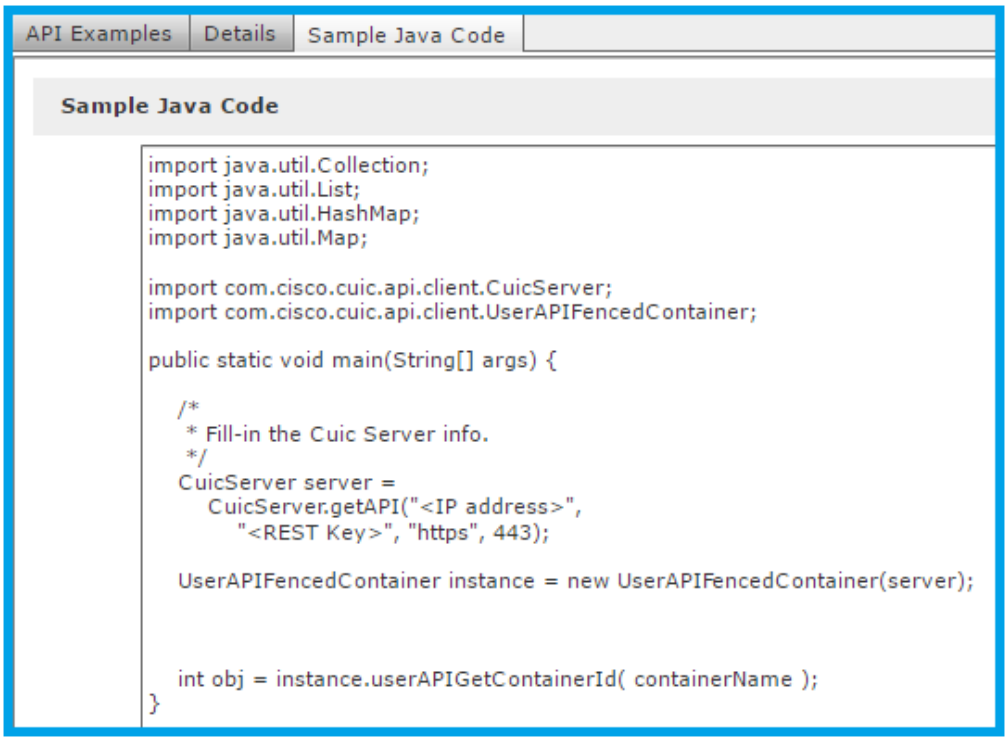
フィールドに情報を入力して [XMLの生成（Generate XML）] をクリックし、グループを作成するためのサンプルの XML ファイルを生成します。



[UCSDサーバ（UCSD Server）] ドロップダウンリストから、API がターゲットとなる Cisco UCS Director を選択し、応答に対し [REST APIの実行（Execute REST API）] をクリックします。

2.1.3 サンプル Java コード

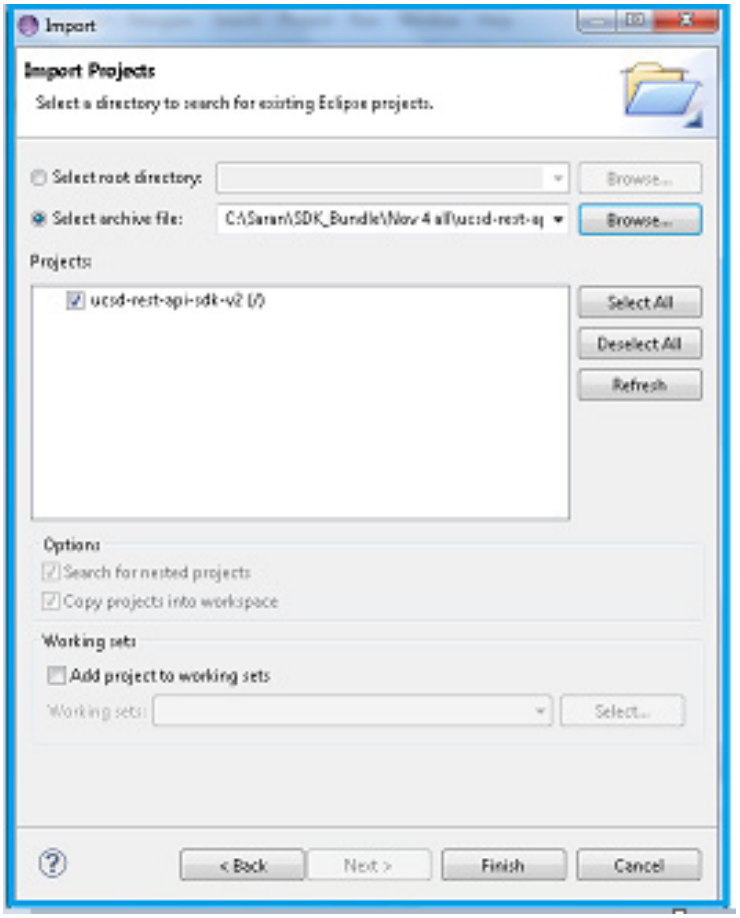
[サンプルJavaコード（Sample Java Code）] タブには、コードによって管理サービスを自動するために使用できるコード スニペットが提示されます。



コード スニペットは、各 API を実行するために使用できます。Eclipse でコードを実行し、出力を取得するには、SDK バンドルを Java プロジェクトとして Eclipse IDE にインポートする必要があります。

SDK バンドルをインポートするには、次の手順を実行します。

1. Eclipse IDE で、[ファイル（File）] > [新規（New）] > [Javaプロジェクト（Java Project）] を選択します。
   [Javaプロジェクトの作成（Create a Java Project）] ダイアログボックスが表示されます。
2. [プロジェクト名（Project Name）] フィールドにプロジェクトの名前を入力します。
3. プロジェクトを右クリックして、[インポート（Import）] を選択します。
4. [インポート（Import）] ダイアログボックスで、[既存プロジェクトをワークスペースへ（Existing projects into Workspace）] を選択し、[次へ（Next）] をクリックします。
5. [参照（Browse）] をクリックして、SDK バンドルを抽出したフォルダにナビゲートします。



6. [終了(Finish)] をクリックします。
Eclipse IDE の [プロジェクトエクスプローラ(Project Explorer)] タブに SDK バンドル プロジェクトが表示されます。

3 API の使用例

ここでは、API 使用時のコード例を示します。各 API の詳細な説明については、『Cisco UCS Director REST API Developer Guide』を参照してください。

3.1 管理(Administration)

3.1.1 グループの作成

次に示すように、グループを作成するための API はいくつかあります。

```
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);
       }
}
```

次に示す別の API のアプローチと使用例は、グループを作成するワークフロー タスクを実行した場合と同等です。

```
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 は、個々のワークフロー タスクの実行に API ラッパーを提供します。この項で示すように、同じジョブを行う複数の API が SDK バンドルで使用可能な場合があります。したがって、開発者は、前述のように、それらの API のいずれかを使用するものとします。

3.1.2 グループのリスト化

```
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 グループの変更

```
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 グループの削除

```
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 ワークフロー オーケストレーション

3.2.1 Service Request の利用

```
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 VApp サービス リクエストの送信

```
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 管理

3.3.1 VM の電源オン

```
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 VM の電源オフ

```
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 VM の再起動

```
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 VM ディスクの作成とサイズ変更

```
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 VMware VM ゲストのセットアップと VIX スクリプトの実行

```
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 VMware スナップショットの削除

```
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 物理アカウントの管理

3.4.1 物理アカウントの作成

```
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 物理アカウントの削除

```
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 アカウントのリスト化

```
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 カタログの管理

3.5.1 カタログ項目の作成

```
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 カタログの詳細の取得

```
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 カタログ項目の削除

```
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 VDC の管理

3.6.1 VDC のリスト化

```
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 VDC の作成

```
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 VDC のエクスポート

```
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 VDC のインポート

```
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 VDC の削除

```
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 コンテナの管理

3.7.1 サービス コンテナの取得

```
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 カタログを使用したサービス コンテナの取得

```
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 サービス コンテナの削除

```
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 );
       }
}
```