Examples

This chapter contains the following sections:

Managing Groups

Creating a Group

Objective

Create a new group with the specified data (group name, description, and contact details) in Cisco UCS Director.

Context

It is mandatory to create a group before adding a service end-user or group admin to Cisco UCS Director. A service end-user or group admin is assigned to a group. Based on the permissions, users within the group will inherit read/write permissions to resources.

Prerequisites

The REST API must be called with an admin user ID.

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPICreateGroup&opData=
{param0:{"groupName":"SDK_Demo_Group","description":"testgroup", "parentGroupId":-1,
"parentGroupName":"any","emailAddress":"sdk@cisco.com", "lastName":"adwin",
"firstName":"Michle","phoneNumber":"2344566","address":"SanJose", "groupType":0,
"enableBudget":true}}
```

Response

```
{ "serviceResult":2, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"userAPICreateGroup" }
```

Components

The parameters of the userAPICreateGroup API are:

- String groupName—The name of the group or organization.
- String description—Optional. The description of the group or organization.
- int parentGroupId—Optional. The ID of the parent group to which the customer group has to be mapped. This parameter is required when the groupType is set as 0 and the MSP mode is enabled in Cisco UCS Director.
- int parentGroupName—Optional. The name of the parent group to which the customer group has to be mapped. This parameter is required when the groupType is set as 0 and the MSP mode is enabled in Cisco UCS Director.

  Note To assign a parent group to a customer group, both the parentGroupId and parentGroupName parameters are mandatory.

- String emailAddress—The email address used to notify the group owner about the status of service requests and request approvals if necessary.
- String lastName—Optional. The last name of the group owner.
- String firstName—Optional. The first name of the group owner.
- String phoneNumber—Optional. The phone number of the group owner.
- String address—Optional. The address of the group owner.
- int groupType—By default, the group type is set as 0. Set the group type as 0 for user groups and customer organizations, and as 1 for Managed Service Provider (MSP) organization user.
- boolean enableBudget—Optional. Set to true to create a group with a budget watch.

Code

JSON-based API

```
import com.cisco.cuic.api.client.APIGroup;
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.UserAPIGlobal;
```

```
public class TestuserAPICreateGroup {
 public static void main(String[] args) throws Exception {
  CuicServer server = CuicServer.getAPI("10.10.110.222",
"6F0063A7F7654561A790EACCE1E8626F", "https", 443);
  UserAPIGlobal instance = new UserAPIGlobal(server);
  int groupId;
  APIGroup apiGroup = new  APIGroup();
  apiGroup.setGroupName("custom_group_2"); //Mandatory
  apiGroup.setEmailAddress("cugroup@cisco.com"); //Mandatory
  apiGroup.setGroupType(0); //Optional- can accept either 0 or 1; by default 0.
  apiGroup.setParentGroupId(2); //Optional
  apiGroup.setParentGroupName("MSP ORG 1"); //Optional
  apiGroup.setFirstName("John"); //Optional
  apiGroup.setLastName("Carry"); //Optional
  apiGroup.setAddress("City-204"); //Optional
  apiGroup.setDescription("This is custom_group_2"); //Optional
  apiGroup.setPhoneNumber("123456789"); //Optional

  groupId = instance.userAPICreateGroup(apiGroup);
  System.out.println("Group Id for the group created: "+groupId);
}
}
```

Alternately, you can use one of the following XML-based APIs:

- `group@CREATE`—To create a customer group.
- `msporg@CREATE` —To create an MSP group.

The `group@CREATE` API includes the following additional parameters:

- GroupCode—A shorter name or code name for the group. This name is used in VM and hostname templates.
- GroupSharePolicyId—The ID of group share policy for the users in this group.
- allowPrivateUsers—Allows resource assignment to users.

```
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.accounts.AddGroupConfig;
import com.cisco.cuic.api.models.accounts.AddGroupConfigResponse;

public class TestgroupCreateXMLAPI {
 public static void main(String[] args) throws Exception {
  CuicServer server = CuicServer.getAPI("172.29.110.222",
"6F0063A7F7654561A790EACCE1E8626F", "https", 443);
  AddGroupConfig instance = new AddGroupConfig(server);
  instance.setGroupName("custom_group_5");//Mandatory
  instance.setGroupDescription("This is custom_group_5"); //Optional
  instance.setParentGroup("2"); //Optional
  instance.setGroupCode("cug_5"); //Optional
  instance.setFirstName("John"); //Optional
  instance.setLastName("Kerry"); //Optional
  instance.setGroupContact("cug_5@cisco.com");//Mandatory
  instance.setAddress("City-401"); //Optional
  instance.setAllowPrivateUsers(true); //Optional
  instance.setGroupSharePolicyId("group_share_policy_1"); //Optional
  instance.setPhone("1234567"); //Optional
  AddGroupConfigResponse groupReponse = instance.execute();
  System.out.println("Group Id for the group created:
"+groupReponse.getOUTPUT_GROUP_ID());
```

```
  }
}
```

Results

A unique group ID is returned after successful creation of a group in the Cisco UCS Director server.

Implementation

You can create a group using the `userAPICreateGroup` API. If you want to create a group using the XML APIs, use the `group@CREATE` API to create a customer group and `msporg@CREATE` to create an MSP group.

See Also

Listing All Groups

Modifying a Group

Deleting a Group

Listing All Groups

Objective

Retrieve all groups in Cisco UCS Director as a list.

Context

Retrieve the list of groups in Cisco UCS Director.

Prerequisites

The requesting user must be authorized to get the list of groups.

REST URL

```
/app/api/rest?formatType=json&opName=userAPIGetGroups&opData={}
```

Components

None

Code

```
public class userAPIGetGroupsExample
  {
   public static void main(String[] args) throws Exception
   {
   CuicServer server = CuicServer.getAPI("192.0.2.207", "1A8DE698E2BF4C0B989476A369F0FC64",
  "https", 443);
   UserAPIGlobal instance = new UserAPIGlobal(server);
   List<APIGroup> groupsList = instance.userAPIGetGroups();
   for (Iterator iterator = groupsList.iterator(); iterator.hasNext();)
     {
     APIGroup apiGroup = (APIGroup) iterator.next();
     System.out.println("Group id = "+apiGroup.getGroupId());
     System.out.println(" Group Name = "+apiGroup.getGroupName());
     }
   }
}
```

Results

The code returns a list of groups in Cisco UCS Director.

Implementation

No implementation required.

See Also

Creating a Group

Modifying a Group

Deleting a Group

Modifying a Group

Objective

Update a group with the given details. You cannot update the ID, name, and type of a group.

Context

Update the contact details, description, and parent group of the group.

Prerequisites

- The logged-in user must have permissions to modify the group.
- The group that you want to update must be available in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPIUpdateGroup&opData={param0:
{"groupId":2,"groupName":"SDKTest","description":"testing","parentGroupId":9,
"parentGroupName":"Test", "emailAddress":"test@cisco.com","lastName":"","firstName":"",
"phoneNumber":"","address":"","groupType":0}}
```

Response

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"userAPIUpdateGroup" }
```

Components

It is mandatory to pass the group name and email address to update the group.

- String groupName—The name of the group or organization.
- String description—Optional. The description of the group or organization.
- int parentGroupId—Optional. The ID of the parent group to which the customer group has to be mapped. This parameter is required when the groupType is set as 0 and the MSP mode is enabled in Cisco UCS Director.
- int parentGroupName—Optional. The name of the parent group to which the customer group has to be mapped. This parameter is required when the groupType is set as 0 and the MSP mode is enabled in Cisco UCS Director.

Note To assign a parent group to a customer group, both the parentGroupId and parentGroupName parameters are mandatory.

- String emailAddress—The email address used to notify the group owner about the status of service requests and request approvals if necessary.
- String lastName—Optional. The last name of the group owner.
- String firstName—Optional. The first name of the group owner.
- String phoneNumber—Optional. The phone number of the group owner.
- String address—Optional. The address of the group owner.
- int groupType—By default, the group type is set as 0. Set the group type as 0 for user groups and customer organizations, and as 1 for Managed Service Provider (MSP) organization user.
- boolean enableBudget—Optional. Set to true to create a group with a budget watch.

Code

```
import com.cisco.cuic.api.client.APIGroup;
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.UserAPIGlobal;

public class TestuserAPIUpdateGroup {
 public static void main(String[] args)throws Exception {
  CuicServer server = CuicServer.getAPI("172.29.110.222",
"6F0063A7F7654561A790EACCE1E8626F", "https", 443);
  boolean isGroupUpdated;
  UserAPIGlobal instance = new UserAPIGlobal(server);
  APIGroup apiGroup = new  APIGroup();
  apiGroup.setGroupName("custom_group_2"); //Mandatory
  apiGroup.setEmailAddress("cugroup@cisco.com"); //Mandatory
  apiGroup.setGroupType(0); //Optional- can accept either 0 or 1; by default 0.
  apiGroup.setParentGroupId(11); //Optional
  apiGroup.setParentGroupName("MSP ORG 2"); //Optional
  apiGroup.setFirstName("John"); //Optional
  apiGroup.setLastName("Carry"); //Optional
  apiGroup.setAddress("City-204"); //Optional
  apiGroup.setDescription("This is custom_group_2"); //Optional
  apiGroup.setPhoneNumber("123456789"); //Optional
  isGroupUpdated = instance.userAPIUpdateGroup(apiGroup);
  System.out.println("Is the Group updated: "+isGroupUpdated);
 }
}
```

Results

If the group is updated successfully, the result is true.

Implementation

Pass the group name and set the necessary group properties that needs to be updated.

See Also

Creating a Group

Listing All Groups

Deleting a Group

Deleting a Group

Objective

Delete a group from Cisco UCS Director based on the given group ID.

Context

Ensure that the user belonging to the group will not access resources in Cisco UCS Director server.

Prerequisites

The group ID must be available in Cisco UCS Director. You can retrieve the group ID using the userAPIGetGroups API.

REST URL

```
/app/api/rest?formatType=json&opName=userAPIDeleteGroup&opData={param0:2}
```

Components

int groupId—The unique ID of the group that needs to be deleted.

Code

```
public class userAPIDeleteGroupExample
{
 public static void main(String[] args) throws Exception
 {
  CuicServer server = CuicServer.getAPI("192.0.2.207", "1A8DE698E2BF4C0B989476A369F0FC64",
 "https", 443);
  UserAPIGlobal instance = new UserAPIGlobal(server);
  //Get the Group id by executing List<APIGroup> groupsList = instance.userAPIGetGroups();

  boolean obj = instance.userAPIDeleteGroup(2);
  System.out.println("is Deleted successfully ? " + obj);
  }
}
```

Results

If the group is deleted successfully, the result is true.

Implementation

Delete a group by passing the group ID.

See Also

Creating a Group

Listing All Groups

Modifying a Group

Managing Users

Resetting the Password of a Logged-in User

Objective

Reset the password of a currently logged in user.

Context

You can use the userAPIModifyLoginProfilePassword API to reset the password of currently logged in user. As the userAPIResetMyPassword API is deprecated, we recommend you to use the userAPIModifyLoginProfilePassword API.

Prerequisites

The oldPassword must be correct.

REST URL

Request

```
/app/api/rest?formatType=json&opName=
userAPIModifyLoginProfilePassword&opData={param0:{"oldPassword":"apadmin1",
"newPassword":"apadmin2"}}
```

Response

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"userAPIModifyLoginProfilePassword" }
```

Components

- String oldPassword—The old password of the currently logged in user.
- String newPassword—The new password for the currently logged in user.

Code

```
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.UserAPIGlobal;
import com.cisco.cuic.api.models.accounts.LoginProfileChangePasswordInfo;
public class TestUserAPIModifyLoginProfilePassword {
 public static void main(String[] args) {
  CuicServer server = CuicServer.getAPI("172.22.234.172",
"50E67524B33F404C80BB8E90D276A560", "https", 443, 12000, "none", "");
  UserAPIGlobal instance = new UserAPIGlobal(server);
  LoginProfileChangePasswordInfo passwordInfo = new LoginProfileChangePasswordInfo();
  passwordInfo.setOldPassword("admin1");
  passwordInfo.setNewPassword("admin2");
  boolean passwordChanged=false;
  try {
   passwordChanged = instance.userAPIModifyLoginProfilePassword(passwordInfo);
   System.out.println("Is Password changed: "+passwordChanged);
  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }
 }
}
```

Results

If the password is reset successfully, the result is true.

Implementation

Call the userAPIModifyLoginProfilePassword API and pass the oldPassword and newPassword to reset the password of currently logged in user.

See Also

Resetting the Password of a User

Resetting the Password of a User

Objective

Reset the password of a user and regenerate the rest API access key of the user.

Context

As an Admin user, you can use the userAPIModifyUserPassword API to reset the password of a user. The REST API access key is also generated for the user if the resetAPIKey parameter is set to true. As the userAPIResetUserPassword API is deprecated, we recommend you to use the userAPIModifyUserPassword API.

Prerequisites

The user whose password is going to be reset, must be a valid Cisco UCS Director user.

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPIModifyUserPassword&opData=
{param0:{"loginUserpassword":"admin5","loginName":"senduser","newPassword":"senduser3",
"resetAPIKey":true}}
```

Response

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"userAPIModifyUserPassword" }
```

Components

- String loginUserpassword—The password of user that needs to be reset.
- String loginName—The name of the user whose password has to be reset.
- String newPassword—The new password of the user.
- boolean resetAPIKey—Set to true to regenerate the REST API access key after resetting the password.

Code

```
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.UserAPIGlobal;
import com.cisco.cuic.api.models.accounts.ModifyUserPasswordInfo;

public class TestUserAPIModifyUserPassword {

 public static void main(String[] args) {
  CuicServer server = CuicServer.getAPI("172.22.234.172",
"91E7A134048F45FD88A29FCBBFD2C233", "https", 443, 12000, "none", "");
```

```
  UserAPIGlobal instance = new UserAPIGlobal(server);
  boolean isPasswordchanged = false;
  ModifyUserPasswordInfo userPasswordInfo = new ModifyUserPasswordInfo();
  userPasswordInfo.setLoginUserpassword("admin5");
  userPasswordInfo.setLoginName("senduser");
  userPasswordInfo.setNewPassword("senduser11");
  userPasswordInfo.setResetAPIKey(true);
  try {
   isPasswordchanged = instance.userAPIModifyUserPassword(userPasswordInfo);
   System.out.println("Is password changed successfully? : "+isPasswordchanged);

  } catch (Exception e) {

   e.printStackTrace();
  }
  }

}
```

Results

If the password is reset successfully, the result is true.

Implementation

Call the userAPIModifyUserPassword API and set the parameters mentioned under components to change the password of a user.

See Also

Resetting the Password of a Logged-in User

Managing Catalogs

Creating a Catalog Item

Objective

Create a catalog item for provisioning a VM, under one of the following catalog types:

- Standard—For creating catalogs for VM provisioning using images from a list of clouds.
- Advanced—For publishing orchestration workflows such as catalog items.
- Service Container—For publishing application containers as catalog items.

Note To create catalogs for Baremetal server provisioning, use the bareMetalCatalog@CREATE API.

The catalog item defines parameters such as the cloud name and the group name to which the VM is bound.

Context

The system administrator creates a catalog item.

Prerequisites

- The cloud, image, and groups must exist in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPICreateCatalogItem&opData=
{param0:{"catalogType":"Standard","catalogItemName":"sample","catalogItemDescription":"sample",
"catalogIcon":"VM: IBM","isAppliedToAllGroups":false,"groups":"Default
Group","publishToEndUsers":true,
"folderName":"Standard","standardCatalog":{"cloudName":"NBTV-Cloud","image":"win8template",
"category":"Generic VM","supportEmail":"tesahu@cisco.com","os":"Linux -
CentOS","otherOS":"sample",
"appLists":"","otherApps":"sample","applicationCode":"","credentialOption":"Do not
share",
"userId":"sample","password":"sample","isAutomaticGuestCustomization":true,"enablePostProvisioningCustomActions":false}}}
```

Response

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
"opName":"userAPICreateCatalogItem" }
```

Components

The parameter of the userAPICreateCatalogItem are:

- catalogType—The type of catalog. It can be one of the following: Advanced, Service Container, and Standard.
- catalogItemName—The name of the catalog item.
- catalogItemDescription—The catalog item description.
- catalogIcon— The catalog icon to associate this catalog with an image. This icon is seen when you are creating a service request using this catalog. For example, VM: IBM,VM: Apache Organization,VM: JBoss Web Server,VM: SugarCRM,VM: Ubuntu Linux,VM: Fedora Linux,VM: SUSE Linux,VM: Redhat Linux,VM: CentOS Linux,VM: Linux,VM: Windows Image 2,VM: Windows Image 1,VM: VMware Image 3,VM: VMware Image 2, and VM: VMware Image.
- isAppliedToAllGroups—Set to true to enable all groups to use this catalog. Set to false to deny its use to other groups.
- groups—The groups that can use this catalog to provision new VMs.
- publishToEndUsers—Set to true if you want to show this catalog to the end users of the system.
- folderName—The folder within which this catalog must be created.
- standardCatalog—The prefix **`standard`** should be the same as catalogType, otherwise, a java.lang.NullPointerException appears.
- cloudName—The cloud with the image for VM provisioning.
- image—The image that you selected from the inventory. For example, window10-template, Windows10, win10, Windows2012-Template, win8template, win7template, Win12template, Centos6.4, and Centos6.8.
- category—The cagory of the catalog. It can be Generic VM, Discovered VM, Web Server, and so on. The details can referred at **Policies** > **Catalog**.

- supportEmail—The email ID of the support person.
- os, otherOS, appLists, otherApps, applicationCode—These parameters refer to the descriptions about the catalog and can be set as empty strings.
- credentialOption—Set one of the following to allow or disallow users to retrieve VM access credentials: Do not share, Share after password reset, and Share template credentials.
- userId—The user ID for VM access.
- password—The password for VM access.
- isAutomaticGuestCustomization—By default, true is set to enable automatic guest customization. If you set this parameter as false, then Cisco UCS Director does not configure the DNS, Network, and Guest OS properties.
- enablePostProvisioningCustomActions—Set to true to enable execution of an orchestration workflow after VM provisioning.

Code

```
import java.util.ArrayList;
import java.util.List;

import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.UserAPIGlobal;
import com.cisco.cuic.api.models.catalog.APICatalogItem;
import com.cisco.cuic.api.models.catalog.StandardCatalogParameters;
import com.cisco.cuic.api.models.catalog.NameValuePair;

public class UserAPICreateCatalogItemSdkSample
{
public static void main(String[] args)
{
 CuicServer server = CuicServer.getAPI("172.29.110.194",
"5AD45F2DC5ED441A9A743F1B219CC302", "http", 80);
 UserAPIGlobal instance = new UserAPIGlobal(server);
 List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();
     StandardCatalogParameters standardCatalogParameters = new
StandardCatalogParameters();
     standardCatalogParameters.setCloudName("vmware117");
     standardCatalogParameters.setImage("CentOSTiny");
     standardCatalogParameters.setCategory("Generic VM");
     standardCatalogParameters.setSupportEmail("sdk@cisco.com");
     standardCatalogParameters.setOs("Windows Server 2012");
     standardCatalogParameters.setOtherOS("Linux Server");
     standardCatalogParameters.setAppLists("Apache Web Server");
     standardCatalogParameters.setOtherApps("Glass Fish web server");
     standardCatalogParameters.setApplicationCode("sdk");
     standardCatalogParameters.setCredentialOption("Do not share");
     standardCatalogParameters.setUserId("admin");
     standardCatalogParameters.setPassword("root");
     standardCatalogParameters.setAutomaticGuestCustomization(false);
     standardCatalogParameters.setEnablePostProvisioningCustomActions(false);
     standardCatalogParameters.setWorkflowName("Print Number");
     standardCatalogParameters.setParameters(nameValuePairs);
 APICatalogItem apiCatalogItem = new APICatalogItem();
 apiCatalogItem.setCatalogType("Standard");
 apiCatalogItem.setCatalogItemName("standardCatalog2");
 apiCatalogItem.setCatalogItemDescription("created through client code");
 apiCatalogItem.setCatalogIcon("VM: SUSE Linux");
 apiCatalogItem.setAppliedToAllGroups(false);
 apiCatalogItem.setGroups("Default Group");
 apiCatalogItem.setPublishToEndUsers(true);
 apiCatalogItem.setFolderName("Standard");
 apiCatalogItem.setStandardCatalog(standardCatalogParameters);
 boolean isCatalogItemCreated = false;
 try {
  isCatalogItemCreated = instance.userAPICreateCatalogItem(apiCatalogItem);
 } catch (Exception e) {
  System.out.error(e.getMessage());
  System.out.error("Exception occured while creating a catalog.");
 }
 System.out.println("Is Catalog Item Got Created ?"+isCatalogItemCreated);
}
}
```

Results

If the catalog item is created successfully, the result is true.

Implementation

Create an APICatalogItem by passing necessary information and then call the userAPICreateCatalogItem API to create the catalog item.

See Also

Retrieving Catalog Details

Deleting a Catalog Item

Retrieving Catalog Details

Objective

Retrieve the details of a catalog using the catalog name.

Context

The catalog details are retrieved by the system administrator or the end user.

Prerequisites

- The requested catalog item must exist.
- The catalog name must be known.

REST URL

```
/app/api/rest?formatType=json&opName=userAPIGetAllCatalogs&opData={}
```

Components

None

Code

```
public class userAPIGetCatalogDetailsExampe
{
 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 params = instance.userAPIGetCatalogDetails("SDKCat");
  System.out.println(" Catalog name "+params.getCatalogName());
  System.out.println(" vDC name "+params.getVdcName());
  }
 }
```

Results

If the catalog details are retrieved successfully, the result is true.

Implementation

Call the userAPIGetCatalogDetails API by passing the catalog name to retrieve the catalog details.

See Also

Creating a Catalog Item

Deleting a Catalog Item

Deleting a Catalog Item

Objective

Delete a catalog item by passing the catalog name.

Context

The catalog item can be deleted by a system administrator.

Prerequisites

The catalog item to be deleted must exist.

REST URL

```
/app/api/rest?formatType=json&opName=userAPIDeleteCatalogItem&opData={param0:"sdkCatalog"}
```

Components

catalogItemName—The name of the catalog item that needs to be deleted.

Code

```
public class userAPIDeleteCatalogItemExample
{
 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 isDeleted = instance.userAPIDeleteCatalogItem("SDKCat");
  System.out.println("is Deleted ?"+isDeleted);
 }
}
```

Results

If the catalog item is deleted successfully, the result is true.

Implementation

Call the userAPIDeleteCatalogItem API by passing the catalog name to delete an existing catalog item.

See Also

Creating a Catalog Item

Retrieving Catalog Details

Managing Physical Accounts

Creating a Physical Account

Objective

Create an account for a physical asset in Cisco UCS Director. When creating an account using the XML-based API, refer to the possible values of the account category and account type parameters in the Components section to pass valid values.

Context

Manage the physical infrastructure resources.

Prerequisites

- A site and pod must exist and reachable.
- The resources of the physical account must be reachable.

REST URL

```
/app/api/rest?formatType=json&opName=accounts:userAPICreateInfraAccount&opData={param0:
{"accountName":"Test Vmware82","podName":"Default Pod","accountCategory":-1,"accountType":
"VMWare","deviceCategory":"","description":"sample","contact":"sample","destinationIPAddress":
"172.29.109.82","login":"administrator","password":"cloupia123","enablePassword":"","protocol":
"https","port":443,"infraAccountSupportDetailsInfo":null}}
```

Components

- InfraAccountDetails infraAccountDetails

  The InfraAccountDetails includes the following parameters:

  - String accountName—A unique name that you assign to this account.
  - String podName—The pod to which this account belongs.
  - int accountCategory—The category of the account. The possible values of the account category are:

Integer	Account Category
1	Compute
2	Storage
3	Network
4	Multi-Domain Manager or Others
5	Cloud

  - String accountType—The type of the account. You need to pass the value within quotes as "11".

The possible values of the account type are:

Integer	Account Type
2	VMWARE
6	HYPERV
9	REDHAT_KVM
10	XENDESKTOP
11	UCSM
12	NETAPP
14	NETAPP_DFM
15	HP
16	CISCO_CIMC
17	EMC VNX File
18	IPMI
19	LOAD_BALANCERS
20	EMC_VMAX
24	EMC VPLEX
22	WHIPTAIL
23	EMC_ISILON
25	EMC_NEW_VNX
26	EMC_NEW_VNX_BLOCK
27	EMC VNX UNIFIED
28	HP_OA
29	CAT_LDAP
30	CAT_LDAP_CLEANUP
31	VCE_VISION_IO
32	EMC_RECOVERPOINT
33	UCS_INVICTA_APPLIANCE
34	UCS_INVICTA_SCALING
35	EMC_NEW_VNX_BLOCK_HTTP

Integer	Account Type
36	EMC_VNXE

- String deviceCategory—The category of the device, including Compute, Storage, and Network.
- String description—Optional. A description of this account.
- String contact—Optional. The email address that you can use to contact the administrator or other person responsible for this account.
- String destinationIPAddress—The destination IP address of this account. You need to pass the IP address within quotes.
- String login—The login ID that this account will use to access element manager. For instance, the Cisco UCS Manager account will use this login ID to access Cisco UCS Manager . This username must be a valid account in Cisco UCS Manager .
- String password—The password associated with the login ID.
- String enablepassword—Optional. Pass the value as true to enable password for this account.
- String protocol—The protocol to use to communicate with the account. The possible values are Telnet and SSH.
- int port—The port that is used to access the element manager.
- infraAccountSupportDetailsInfo—Details of the infra account. The infraAccountSupportDetailsInfo includes the following parameters:
  - String spAIpAddress—Optional. The IP address for Storage Processor A of the VNX device.
  - String spBIpAddress—Optional. The IP address for Storage Processor B of the VNX device.
  - String blockAccessUserName—Optional. The username for block access of the VNX device.
  - String blockAccessPwd—Optional. The password for block access of the VNX device.
  - String sshIpAddress—Optional. The IP address of the SSH server. You need to pass the IP address within quotes.
  - String sshUsername—Optional. The username that this account will use to access the SSH server.
  - String sshPassword—Optional. The password associated with the SSH username.
  - int sshPort—Optional. The port used to access the SSH server.
  - String domain—Optional. The domain that is associated with the account.
  - String serviceProvider—Optional. The name of the service provider associated with this account, if any.

Code

```
public class UserAPICreateInfraAccountExample
{
 public static void main(String[] args) throws Exception
 {
  CuicServer server = CuicServer.getAPI("192.0.2.187", "6C6416AD90704DF495A6B4D0A75A0BB1",
 "https", 443);
  UserAPIAccounts instance = new UserAPIAccounts(server);
  InfraAccountDetails infraAccountDetails = new InfraAccountDetails();
  infraAccountDetails.setAccountName("AccName");
  infraAccountDetails.setPodName("PodName");
  infraAccountDetails.setAccountCategory(2);
  infraAccountDetails.setAccountType("11");
  infraAccountDetails.setProtocol("http");
  infraAccountDetails.setPort(80);
  infraAccountDetails.setDestinationIPAddress("172.29.32.14");
  infraAccountDetails.setLogin("admin");
  infraAccountDetails.setPassword("admin");
  boolean isCreated = instance.userAPICreateInfraAccount(infraAccountDetails);
  System.out.println("is Account Created ?"+isCreated);
 }
}
```

Results

A physical account is created. The return value is true if creation is successful.

Implementation

Create an instance of InfraAccountDetails object with account information and then call userAPICreateInfraAccount to create the physical account.

See Also

Listing the Accounts
Deleting a Physical Account

Listing the Accounts

Objective

Retrieve the physical and virtual accounts in Cisco UCS Director.

Context

Identify the existing resources or accounts in Cisco UCS Director.

Prerequisites

User must have permission to view all the accounts in Cisco UCS Director.

REST URL

```
/app/api/rest?formatType=json&opName=accounts:userAPIGetAllPhysicalInfraAccounts&opData={}
```

Components

None

Code

```
public class UserAPIGetAllAccountsExample
{
 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> accountNameList = instance.userAPIGetAllAccounts();
  for (int i = 0; i < accountNameList.size(); ++i)
   {
   System.out.println("Name "+accountNameList.get(i));
   }
  }
}
```

Results

The list of physical and virtual accounts is displayed.

Implementation

To retrieve the existing physical and virtual account details, call the userAPIGetAllAccounts API.

See Also

Creating a Physical Account

Deleting a Physical Account

Deleting a Physical Account

Objective

Delete a physical account from Cisco UCS Director.

Context

The physical account can be deleted by a user belonging to the group to which the account is mapped.

Prerequisites

The physical account must be available.

REST URL

```
/app/api/rest?formatType=json&opName=accounts:userAPIDeleteInfraAccount&opData={param0:"UCSM-150"}
```

Components

String Account name—The name of the physical account that needs to be deleted.

Code

```
public class UserAPIDeleteInfraAccountExample
{
 public static void main(String[] args) throws Exception
 {
 CuicServer server = CuicServer.getAPI("192.0.2.207", "1A8DE698E2BF4C0B989476A369F0FC64",
"https", 443);
  UserAPIAccounts instance = new UserAPIAccounts(server);
  boolean isDeleted = instance.userAPIDeleteInfraAccount("UCSAccount");
  System.out.println("Is the Account deleted ? :" + isDeleted);
 }
}
```

Results

If the physical account is deleted successfully, the result is true.

Implementation

Call the userAPIDeleteInfraAccount API by passing the physical account name to delete an existing physical account.

See Also

Creating a Physical Account

Listing the Accounts

Managing Virtual Data Centers

Creating a VDC

Objective

Create a virtual data center (VDC) from which the VM will be provisioned.

Context

Create a VDC to provision a VM.

Prerequisites

Cloud and group must exist.

REST URL

```
/app/api/rest?formatType=json&opName=userAPICreateVDC&opData={param0:{"vdcName":"Vdc"
,"vdcDescription":"VDC","cloudName":"VMware-Cloud","groupName":2,"approver1":"","approver2":
"","vdcSupportEmail":"test@cisco.com","vdcCustomerNoticationEmail":"","systemPolicy":
"VmwareCloudSysPolicy","deploymentPolicy":"","slaPolicy":"","computingPolicy":"","storagePolicy":
"","networkPolicy":"","costModel":"","isLocked":false,"isDeletable":false,
"inactivityPeriodForDeletion":1000}}
```

Components

APIVDCDetails

Code

```
public class CreateVDCExample
 {
 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("Vdc");
  vdcDetails.setVdcDescription("VDC");
  vdcDetails.setCloudName("VMware-Cloud");
  vdcDetails.setGroupName(2);
  vdcDetails.setApprover1("");
  vdcDetails.setApprover2("");
  vdcDetails.setVdcSupportEmail("test@cisco.com");
  vdcDetails.setVdcCustomerNoticationEmail("");
  //System policy is optional
  vdcDetails.setSystemPolicy("VmwareCloudSysPolicy");
  //System policy is optional
  vdcDetails.setSlaPolicy("");
  //System policy is optional
  vdcDetails.setComputingPolicy("");
  //netowrk policy is optional
  vdcDetails.setNetworkPolicy("");
  //storage  policy is optional
  vdcDetails.setStoragePolicy("");
  //Cost model is optional
  vdcDetails.setCostModel("");
  //vdc locked  is optional
  vdcDetails.setLocked(false);
  //Deletable is optional
  vdcDetails.setDeletable(false);
  vdcDetails.setInactivityPeriodForDeletion(1000);
  boolean isVDCCreated = instance.userAPICreateVDC(vdcDetails);
  System.out.println("is VDC Created "+isVDCCreated);
 }
}
```

Results

If the VDC is created successfully, the result is true.

Implementation

Create an APIVDCDetails by passing necessary information and then call the userAPICreateVDC to create a VDC.

See Also

Listing the VDCs

Exporting a VDC

Importing a VDC

Retrieving VDC Resource Limits, on page 24

Retrieving a Cost Model, on page 25

Deleting a VDC

Listing the VDCs

Objective

Retrieve a list of VDCs in a user group.

Context

Retrieve a list of VDCs in a group to choose the required VDC when provisioning VMs.

Prerequisites

The logged-in user must be assigned to the group.

REST URL

```
/app/api/rest?formatType=json&opName=userAPIGetAllVDCs&opData={}
```

Note After upgrading from 5.3 or earlier releases of Cisco UCS Director, if you get NullPointerException response for the UserAPIGlobal@userAPIGetAllVDCs API, check if the null value is displayed in the header section of the MSP Organization report or Customer Organization report. If the null value is displayed, do the following:

1. In Cisco UCS Director, choose **Administration** > **System**.
2. Click **Service Provider Feature**.
3. Click **Submit**.

Components

None

Code

```
public class UserAPIGetAllVDCExample
{
 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 tabularReport = instance.userAPIGetAllVDCs();
  System.out.println("No. Of VDC :"+tabularReport.getRowCount());
  System.out.println("No. Of Column : "+tabularReport.getColumnMetaData());
  }
}
```

Results

The list of VDCs in the user group is returned.

Implementation

Call the userAPIGetAllVDCs API to retrieve all the VDCs in a user group.

See Also

Creating a VDC

Exporting a VDC

Importing a VDC

Retrieving VDC Resource Limits, on page 24

Retrieving a Cost Model, on page 25

Deleting a VDC

Exporting a VDC

Objective

Export a VDC from the Cisco UCS Director server to the local system.

Context

Create a same set of VDC in another Cisco UCS Director server.

Prerequisites

The name of the VDC that you want to export.

REST URL

```
/app/api/rest?formatType=json&opName=userAPIExportVDC&opData={param0:"vDCIT"}
```

Components

vdcName—The name of the VDC that you want to export.

Code

```
public class ExportVDCExample
 {
 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 exportVDCString = instance.userAPIExportVDC("sample_VDC");
  System.out.println("Export VDC as "+exportVDCString);
 }
}
```

Results

The sample_VDC VDC is exported to local system.

Implementation

Use the userAPIExportVDC API and pass the VDC name to export the VDC to the local system.

See Also

Listing the VDCs

Creating a VDC

Importing a VDC

Retrieving VDC Resource Limits, on page 24

Retrieving a Cost Model, on page 25

Deleting a VDC

Importing a VDC

Objective

Import a VDC into Cisco UCS Director from the local system.

Context

Import external VDC into Cisco UCS Director.

Prerequisites

The VDC that needs to be imported must be available in the local system.

REST URL

```
/app/api/rest?formatType=json&opName=userAPIImportVDC&opData={param0:"importvdcasstring"}
```

Components

vdcName—The name of the VDC that you want to import.

Code

```
public class ImportVDCExample
{
 public static void main(String[] args) throws Exception
 {
 CuicServer server = CuicServer.getAPI("192.0.2.207",  "1A8DE698E2BF4C0B989476A369F0FC64",
 "https", 443);
  UserAPIGlobal instance = new UserAPIGlobal(server);
  //You have to pass the exported VDC String as importVDC parameter
  VDC vdcObject = instance.userAPIImportVDC("exportVDCAsString" );
  System.out.println("VDC Name = "+vdcObject.getVdcName());
 }
}
```

Results

The imported VDC appears in Cisco UCS Director.

Implementation

Call the userAPIImportVDC API and pass the VDC name to import the VDC into Cisco UCS Director.

See Also

Creating a VDC

Listing the VDCs

Exporting a VDC

Retrieving VDC Resource Limits, on page 24

Retrieving a Cost Model, on page 25

Deleting a VDC

Retrieving VDC Resource Limits

Objective

Retrieve the resource limits of a VDC.

Context

Identify the used resource limits of a VDC.

Prerequisites

Container or VDC must be defined.

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPIGetVDCResourceLimits&opData=
{param0:"bmtest"}
```

Response

```
{ "serviceResult":{"provisionedNoOfvCPUsLimit":0,"provisionedMemGBLimit":0.0,
"provisionedDiskGBLimit":0.0,"halfWidthPhysicalServerLimit":0,"fullWidthPhysicalServerLimit":0},

 "serviceError":null, "serviceName":"InfraMgr",
"opName":"fenced:userAPIGetVDCResourceLimits" }
```

Components

None

Code

```
public class GetVDCResourceLimits
{
 public static void main(String[] args) throws Exception
 {
  CuicServer api = CuicServer.getAPI("172.29.110.241", "EDDF69C4D9AA4E4FA8F14EFD57B90402",
 "http", 80);
  UserAPIFencedContainer fenced = new UserAPIFencedContainer(api);
  APIResourceLimitParams params = new APIResourceLimitParams();
  params.setVdcName("bmtest");
  APIResourceLimitResponse response = fenced.userAPIGetVDCResourceLimits(params);
  System.out.println( " Response is \n Full Width
Limits"+response.getFullWidthPhysicalServerLimit());
  System.out.println( " Half Width Limits"+response.getHalfWidthPhysicalServerLimit());

  System.out.println( " Provisioned Disk GB "+response.getProvisionedDiskGBLimit());
  System.out.println( " Provisioned Memory"+response.getProvisionedMemGBLimit());
  System.out.println( " Provisioned vCPU Limits
"+response.getProvisionedNoOfvCPUsLimit());
 }
}
```

Results

The following resource limits of the VDC are displayed: full width limits, half width limits, provisioned disk, provisioned memory, and provisioned vCPU limits.

Implementation

Use the userAPIGetVDCResourceLimits API and pass the VDC name to view the resource limits configuration.

See Also

Creating a VDC, on page 20

Listing the VDCs

Exporting a VDC

Importing a VDC

Retrieving a Cost Model, on page 25

Deleting a VDC

Retrieving a Cost Model

Objective

Retrieve the cost model for the resources available in the VDC.

Context

Retrieve the one time and monthly cost model of the VM and physical server for the given resources (for example, CPU, memory, and disk).

Prerequisites

VDC and cost model must exist.

REST URL

Request

```
/app/api/rest?formatType=json&opName=chargeback:userAPIGetCostModel&opData=
{param0:{"vdcName":"CostModel_Vdc","costModelResources":[{"name":"CPU","value":"1"},
{"name":"Memory","value":"1"},{"name":"Disk","value":"1"},{"name":"NetworkRx",
"value":"1"},{"name":"NetworkTx","value":"1"},{"name":"BMCPU","value":"1"},
{"name":"BMMemory","value":"1"},{"name":"BMDisk","value":"1"},
{"name":"BladeType","value":"Half"}]}}
```

Response

```
{ "serviceResult":{"costModelRequestedInfo":{"vDC Name":"CostModel_Vdc",
"Memory (GB)":"1","Disk (GB)":"1","NetworkRx (GB)":"1","NetworkTx (GB)":"1",
"CPU (GHz)":"1","BMMemory (GB)":"1","BMDisk (GB)":"1","Blade Type":"Half",
"BMCPU (Cores)":"1"},"vmCostModel":{"oneTimeItemCost":100.0,"monthlyCost":2180.0,
"cpuCostModel":{"cpuGhzCostPerHour":1.0,"cpuCostPerCore":0.0,"totalCPUCost":720.0},
"diskCostModel":{"diskGBCostPerHour":1.0,"totalDiskCost":720.0},"memoryCostModel":
{"memoryGBCostPerHour":1.0,"totalMemoryCost":720.0},"networkCostModel":
{"netRxCostPerGB":10.0,"netTxCostPerGB":10.0,"totalNetworkCost":20.0}},
"bMCostModel":{"oneTimeItemCost":100.0,"monthlyCost":2880.0,"cpuCostModel":
{"cpuGhzCostPerHour":0.0,"cpuCostPerCore":1.0,"totalCPUCost":720.0},
"diskCostModel":{"diskGBCostPerHour":1.0,"totalDiskCost":720.0},"memoryCostModel":
{"memoryGBCostPerHour":1.0,"totalMemoryCost":720.0},"bladeCostModel":
{"fullBladeCostPerHour":0.0,"halfBladeCostPerHour":1.0,"totalBladeCost":720.0}}},
"serviceError":null, "serviceName":"InfraMgr", "opName":"chargeback:userAPIGetCostModel"
 }
```

Components

- VdcName—The name of the VDC.

The possible name:value pairs are:

- CPU—*Optional*. The provisioned CPU limit in GHz or cores.
- Memory—*Optional*. The provisioned memory limit in GB.
- NetworkRx—*Optional*. The rate at which the traffic data is received.
- NetworkTx—*Optional*. The rate at which the traffic data is transmitted.
- Disk—*Optional*. The provisioned disk limit in GB.
- BMCPU—*Optional*. The provisioned CPU limit of BM in GHz or cores.
- BMMemory—*Optional*. The provisioned BM memory limit in GB.
- BMDisk—*Optional*. The provisioned disk limit of BM in GB.
- BladeType—*Optional*. The type of the blade.

Code

```
public class GetCostModel
{
 public static void main(String[] args)throws Exception
{
  CuicServer server = CuicServer.getAPI("<IP address>",
            "<REST Key>", "https", 443);
  UserAPIChargeBack instance = new UserAPIChargeBack(api);
  List<APINameValue> requestParam = new ArrayList<APINameValue>();
  APIGetCostModelParams costParams = new APIGetCostModelParams();
  costParams.setVdcName("CostModel_Vdc");
  APINameValue value=new APINameValue();

  value.setName("CPU");
  value.setValue("1");
  requestParam.add(value);
  value=new APINameValue();
  value.setName("Memory");
  value.setValue("1");
  requestParam.add(value);
  value=new APINameValue();
  value.setName("Disk");
  value.setValue("1");
  requestParam.add(value);
  value=new APINameValue();
  value.setName("NetworkRx");
  value.setValue("1");
  requestParam.add(value);
  value=new APINameValue();
  value.setName("NetworkTx");
  value.setValue("1");
  value=new APINameValue();
  value.setName("BMCPU");
  value.setValue("1");
  requestParam.add(value);
   costParams.setCostModelResources(requestParam);
   APIGetCostModelResponse response = instance.userAPIGetCostModel(costParams);
   APIVMCostModel vmCostModel = response.getVmCostModel();
   APIPhysicalServerCostModel bmCostModel = response.getBMCostModel();
   if(vmCostModel != null){
    System.out.println(vmCostModel.getOneTimeItemCost());
   APICPUCostModel cpuModel =  vmCostModel.getCpuCostModel();
    System.out.println(cpuModel.getTotalCPUCost());
   }
  if(bmCostModel != null){
   System.out.println(bmCostModel.getOneTimeItemCost());
   APICPUCostModel bmCPUModel = bmCostModel.getCpuCostModel();
         System.out.println(bmCPUModel.getTotalCPUCost());
  }
 }
 }
```

Results

The cost model of resources in VM and physical server are displayed.

Implementation

Call the userAPIGetCostModel API and pass the VDC name along with the resources for which you want to see the cost model.

See Also

Creating a VDC, on page 20

Listing the VDCs, on page 21

Exporting a VDC, on page 22

Importing a VDC, on page 23

Retrieving VDC Resource Limits, on page 24

Deleting a VDC, on page 28

Deleting a VDC

Objective

Delete a VDC from Cisco UCS Director.

Context

Remove a VDC which is not in use, from Cisco UCS Director.

Prerequisites

The logged-in user must have permission to delete a VDC.

REST URL

```
Not Applicable
```

Components

vdcName—The name of the VDC that you want to delete.

Code

```
public class DeleteVDCExample
{
 public static void main(String[] args) throws Exception
{
  CuicServer server = CuicServer.getAPI("192.0.2.207", "1A8DE698E2BF4C0B989476A369F0FC64",
 "https", 443);
  DeleteVDCConfig instance = new DeleteVDCConfig(server);
  instance.setVdcName("");
  instance.setRollbackRequest(false);
  boolean isDeleted = instance.execute();
  System.out.println(" is VDC Deleted "+isDeleted);
 }
}
```

Results

If the VDC is deleted successfully, the result is true.

Implementation

Delete a VDC by passing the VDC name.

See Also

Creating a VDC

Listing the VDCs

Exporting a VDC

Importing a VDC

Retrieving VDC Resource Limits, on page 24

Retrieving a Cost Model, on page 25

Managing Virtual Infrastructure Policies

Creating a Virtual Infrastructure Policy

Objective

Create a virtual infrastructure policy for a fenced virtual application container. The virtual infrastructure policy defines which VM to use and what type of container you want to provision. This policy also defines which PNSC account you want to tie to this particular account.

Context

The virtual infrastructure policy is used for creating application container template. With this template, you can create application containers for use in a variety of networks (including DFA Networks).

Prerequisites

The VMware virtual account must exist.

REST URL

Request

```
/app/api/rest?formatType=json&opName=fenced:
userAPICreateServiceContainerVirtualInfraPolicy&opData={param0:{"policyName":
"vipFenc","policyDescription":"","virtualAccountName":"vc117",
"isGatewayRequired":false,"gatewayPolicyName":"","isF5LoadBalancerRequired":
false,"f5LoadBalancerPolicyName":""}}
```

Response

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"fenced:userAPICreateServiceContainerVirtualInfraPolicy" }
```

Components

- policyName—The name of the virtual infrastructure policy.
- policyDescription—*Optional*. The description of the virtual infrastructure policy.
- virtualAccountName—The name of the virtual account for which you define the virtual infrastructure policy.
- isGatewayRequired—Set the parameter as **True** if gateway is required for the fenced container.
- gatewayPolicyName—If the isGatewayRequired parameter is set as **True**, provide the gateway policy name.
- isF5LoadBalancerRequired—Set the parameter as **True** if the F5 load balancer service is required.
- f5LoadBalancerPolicyName—If the isF5LoadBalancerRequired parameter is set as **True**, provide the F5 load balancer policy name.

Code

```
public class FencedContainerVirtualInfraPolicyCreateExample
{
 public static void main(String[] args) throws Exception
{
   CuicServer server =
CuicServer.getAPI("172.22.234.33","EF0FE312B5444E7B8DE4836DDC55F4A0", "https", 443);
   UserAPIFencedContainer instance = new UserAPIFencedContainer(server);
   FencedContainerVirtualInfrastructurePolicy policy = new
FencedContainerVirtualInfrastructurePolicy();
   policy.setPolicyName("VIPolicy");
   policy.setPolicyDescription("VIPolicy1 Description");
   policy.setVirtualAccountName("vm117");
   policy.setGatewayRequired(false);
   policy.setF5LoadBalancerRequired(false);
   boolean obj = instance.userAPICreateServiceContainerVirtualInfraPolicy( policy );
   System.out.println("Created Sucessfuly : "+obj);
 }
}
```

Results

Returns True after successful addition of the virtual infrastructure policy.

Implementation

Call the userAPICreateServiceContainerVirtualInfraPolicy API and pass the virtual account name along with the necessary details.

See Also

Retrieving a Virtual Infrastructure Policy

Modifying a Virtual Infrastructure Policy

Deleting a Virtual Infrastructure Policy

Retrieving a Virtual Infrastructure Policy

Objective

Retrieve the details of a virtual infrastructure policy by passing the policy name.

Context

To view the set policy details.

Prerequisites

The virtual infrastructure policy must exist.

REST URL

Request

```
/app/api/rest?formatType=json&opName=fenced:
userAPIGetServiceContainerVirtualInfraPolicy&opData={param0:"vipPolicy"}
```

Response

```
{ "serviceResult":[{"policyId":2,"policyName":" vipPolicy ",
"policyDescription":"","containerType":"Fenced Virtual","account":"V100",
"vnmcConfig":{"vnmcAccount":null,"vnmcFirewallPolicies":null},"apicConfig":
{"applicationProfile":null},"gatewayConfig":{"gatewayRequired":false,
"gatewayPolicy":"","summaryArea":null,"gatewayType":"No Gateway"},"f5LBConfig":
```

```
{"f5LoadBalancerRequired":false,"f5LoadBalancerPolicy":"","summaryArea":null,
"f5LoadBalancerType":"No F5 Load Balancer"},"dfaConfig":{"vsgEnabled":false,
"accountId":"","switchType":"","switchName":"","alternateSwitchName":"",
"serviceNetworkConfiguration":null,"l3Network":false,"fabricOrganization":"",
"fabricPartition":"","fabricNetwork":null,"serviceNetworkMobilityDomain":null,
"autoServiceNetworkMobilityDomain":true,"hostNetworkConfiguration":null,
"mobilityDomain":null,"autoSelectMobilityDomain":true,"partitionParamsLabel":null,
"dciID":null,"extendPrtnAcrossFabric":true,"serviceNodeIpAddress":null,
"dnsServer":null,"secondaryDNSServer":null,"multiCastGroupAddress":null,
"profileName":"None","profileParamsLabel":null,"includeBorderLeafRt":null},
"fabricASAConfig":{"ipSubnetPoolPolicy":null,"internalConfigurationLabel":null,
"internalNetworkName":null,"internalNetworkRole":null,
"internalNetworkGatewayIP":null,"internalNetworkMask":null,
"internalNetworkProfile":null,"internalNetworkMobilityDomain":null,
"autoSelectInternalNetworkMobilityDomain":true,"externalConfigurationLabel":null,
"externalPartitionProfile":null,"externalNetworkName":null,"externalNetworkRole":null,
"externalNetworkGatewayIP":null,"externalNetworkMask":null,
"externalNetworkProfile":null,"externalNetworkMobilityDomain":null,
"autoSelectExternalNetworkMobilityDomain":true},"fabricASAvConfig":
{"ipSubnetPoolPolicy":null,"internalConfigurationLabel":null,"internalNetworkName":null,
"internalNetworkRole":null,"internalNetworkGatewayIP":null,"internalNetworkMask":null,
"internalNetworkProfile":null,"internalSwitchType":"","internalSwitchName":
"","internalNetworkMobilityDomain":null,"autoSelectInternalNetworkMobilityDomain":true,
"externalConfigurationLabel":null,"externalPartitionProfile":null,"externalNetworkName":null,
"externalNetworkRole":null,"externalNetworkGatewayIP":null,"externalNetworkMask":null,
"externalNetworkProfile":null,"externalSwitchType":"","externalSwitchName":
"","externalNetworkMobilityDomain":null,"autoSelectExternalNetworkMobilityDomain":true},
"fabricF5Config":{"serviceConfigurationLabel":null,"serviceNetworkName":null,
"serviceNetworkRole":null,"serviceNetworkGatewayIP":null,"serviceNetworkMask":null,
"serviceNetworkProfile":null,"f5ServiceNetworkMobilityDomain":null,
"autoSelectServiceNetworkMobilityDomain":true}}], "serviceError":null,
 "serviceName":"InfraMgr", "opName":"fenced:userAPIGetServiceContainerVirtualInfraPolicy"
 }
```

Components

policyName—The name of the virtual infrastructure policy.

Code

```
public class retrieveAPI
{
public static void main(String[] args) throws Exception
{
 CuicServer server = CuicServer.getAPI("10.23.210.118","ADFGLKJSHFJ23478234HJBFJGH",
"https", 443);
 UserAPIFencedContainer instance = new UserAPIFencedContainer(server);
 List obj = instance.userAPIGetServiceContainerVirtualInfraPolicy( "x" );
 System.out.println(obj);
 }
}
```

Results

If the policy details are retrieved successfully, the result is true.

Implementation

Call the userAPIGetServiceContainerVirtualInfraPolicy API and pass the virtual infrastructure policy name.

See Also

Creating a Virtual Infrastructure Policy

Modifying a Virtual Infrastructure Policy

Deleting a Virtual Infrastructure Policy

Modifying a Virtual Infrastructure Policy

Objective

Update the virtual infrastructure policy of a fenced container with the given details. The policy name cannot be edited.

Context

To update the policy of virtual infrastructure defined for a container. The updated policy is used for creating a new application container template.

Prerequisites

The virtual infrastructure policy must exist.

REST URL

Request

```
/app/api/rest?formatType=json&opName=fenced:
userAPIUpdateServiceContainerVirtualInfraPolicy&opData={param0:
{"policyName":"Testing","policyDescription":"Testing Update Description ",
"virtualAccountName":"VNX_Cloud169","isGatewayRequired":false,
"gatewayPolicyName":"","isF5LoadBalancerRequired":false,"f5LoadBalancerPolicyName":""}}
```

Response

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"fenced:userAPIUpdateServiceContainerVirtualInfraPolicy" }
```

Components

- policyName—The name of the virtual infrastructure policy.
- policyDescription—*Optional*. The description of the virtual infrastructure policy.
- virtualAccountName—The name of the virtual account for which you define the virtual infrastructure policy.
- isGatewayRequired—Set the parameter as **True** if gateway is required for the fenced container.
- gatewayPolicyName—If the isGatewayRequired parameter is set as **True**, provide the gateway policy name.
- isF5LoadBalancerRequired—Set the parameter as **True** if the F5 load balancer service is required.
- f5LoadBalancerPolicyName—If the isF5LoadBalancerRequired parameter is set as **True**, provide the F5 load balancer policy name.

Code

```
public class FencedContainerVirtualInfraPolicyUpdateExample
{
 public static void main(String[] args) throws Exception {
 CuicServer server = CuicServer.getAPI("172.22.234.33","EF0FE312B5444E7B8DE4836DDC55F4A0",
 "https", 443);
   UserAPIFencedContainer instance = new UserAPIFencedContainer(server);
   FencedContainerVirtualInfrastructurePolicy policy = new
FencedContainerVirtualInfrastructurePolicy();
   policy.setPolicyName("VIPolicy");
```

```
    policy.setPolicyDescription("VIPolicy1 Description modified");
    policy.setVirtualAccountName("vm117");
    policy.setGatewayRequired(false);
    policy.setF5LoadBalancerRequired(false);
    boolean obj = instance.userAPIUpdateServiceContainerVirtualInfraPolicy( policy );
    System.out.println("Created Sucessfuly : "+obj);
  }
 }
```

Results

If the virtual infrastructure policy is updated successfully, the result is true.

Implementation

Call the userAPIUpdateServiceContainerVirtualInfraPolicy API, and pass the virtual infrastructure policy name and virtual account name along with the necessary details that need to be updated.

See Also

Creating a Virtual Infrastructure Policy

Retrieving a Virtual Infrastructure Policy

Deleting a Virtual Infrastructure Policy

Deleting a Virtual Infrastructure Policy

Objective

Delete a virtual infrastructure policy from Cisco UCS Director.

Context

A user belonging to a group, can delete the virtual infrastructure policy.

Prerequisites

The virtual infrastructure policy must exist.

REST URL

Request

```
/app/api/rest?formatType=json&opName=fenced:
userAPIDeleteServiceContainerVirtualInfraPolicy &opData={param0:{"policyName":"vipFenc”}}
```

Response

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"fenced:userAPIDeleteServiceContainerVirtualInfraPolicy" }
```

Components

- policyName—The name of the virtual infrastructure policy.

Code

```
public class DeleteServiceContainerVirtualInfraPolicy
{
 public static void main(String[] args) throws Exception {
 CuicServer server = CuicServer.getAPI("172.22.234.33","EF0FE312B5444E7B8DE4836DDC55F4A0",
 "https", 443);
 UserAPIFencedContainer instance = new UserAPIFencedContainer(server);
 FencedContainerVirtualInfrastructurePolicy policy = new
FencedContainerVirtualInfrastructurePolicy();
```

```
 policy.setPolicyName("VIPolicy");
 boolean obj = instance.userAPIDeleteServiceContainerVirtualInfraPolicy( policy );
 System.out.println(obj);
 }
}
```

Results

If the virtual infrastructure policy is deleted successfully, the result is true.

Implementation

Call the userAPIDeleteServiceContainerVirtualInfraPolicy API and pass the virtual infrastructure policy name.

See Also

Creating a Virtual Infrastructure Policy

Retrieving a Virtual Infrastructure Policy

Modifying a Virtual Infrastructure Policy

Managing APIC Virtual Infrastructure Policies

Creating an APIC Virtual Infrastructure Policy

Objective

Create a virtual infrastructure policy for an APIC container.

Context

The virtual infrastructure policy is used for creating a service container template.

Prerequisites

The virtual account must exist.

REST URL

Request

```
/app/api/rest?formatType=json&opName=
apic:userAPICreateServiceContainerVirtualInfraPolicy&opData={param0:
{"policyName":"VIPolicy1","policyDescription":"Create VIP Policy","containerType":
"APIC","applicationProfileName":" appprofile"}}
```

Response on successful execution of the API

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
"opName":"apic:userAPICreateServiceContainerVirtualInfraPolicy" }
```

Response on unsuccessful execution of the API

If the application profile specified in the REST URL does not exist, the API throws the following exception. The exception suggests user to use a valid application profile name or to use the userAPICreateApplicationProfile API to create a new application profile.

```
{ "serviceResponse":null, "serviceError":"REMOTE_SERVICE_EXCEPTION:
Application profile test does not exist, Please use userAPICreateApplicationProfile
api to create new application profile.", "serviceName":"InfraMgr",
"opName":"apic:userAPICreateServiceContainerVirtualInfraPolicy" }
```

Components

- policyName—The name of the virtual infrastructure policy.
- containerType—The type of the container must be APIC.
- applicationProfileName—The name of the application profile that you want to use for creating the container.

Code

```
public class APICContainerVirtualInfraPolicyCreateExample
{
 public static void main(String[] args) throws Exception {
 CuicServer server = CuicServer.getAPI("172.22.234.33","EF0FE312B5444E7B8DE4836DDC55F4A0",
 "https", 443);
  UserAPIAPICContainer instance = new UserAPIAPICContainer(server);
  APICContainerVirtualInfraStructurePolicy policy = new
APICContainerVirtualInfraStructurePolicy();
  policy.setPolicyName("VIPolicy1");
  policy.setPolicyDescription("VIPolicy1 Description");
  policy.setApplicationProfileName("applicationProfileName");
  boolean obj = instance.userAPICreateServiceContainerVirtualInfraPolicy(policy);
  System.out.println("Created Sucessfuly : " + obj);

  /**
   * Update the service Container Virtual Infra Policy
   */

  obj = instance.userAPIUpdateServiceContainerVirtualInfraPolicy(policy);
  System.out.println(obj); }
}
```

Results

If the virtual infrastructure policy is added successfully, the result is true.

Implementation

Call the userAPICreateServiceContainerVirtualInfraPolicy API and pass the application profile name along with the necessary details.

See Also

Listing All APIC Virtual Infrastructure Policy

Retrieving an APIC Virtual Infrastructure Policy

Modifying an APIC Virtual Infrastructure Policy

Deleting an APIC Virtual Infrastructure Policy

Listing All APIC Virtual Infrastructure Policy

Objective

Retrieve all APIC virtual infrastructure policies in Cisco UCS Director.

Context

To get a list of APIC virtual infrastructure policies in Cisco UCS Director.

Prerequisites

The requesting user must be authorized to get the list of APIC virtual infrastructure policies.

REST URL

```
/app/api/rest?formatType=json&opName=
apic:userAPIGetAllServiceContainerVirtualInfraPolicies&opData={}
```

Components

None

Code

```
public class GetAllAPICContainerVirtualInfraPolicies
{
 public static void main(String[] args) throws Exception {
 CuicServer server = CuicServer.getAPI("172.22.234.33","EF0FE312B5444E7B8DE4836DDC55F4A0",
 "https", 443);
 UserAPIAPICContainer instance = new UserAPIAPICContainer(server);
  List<APICContainerVirtualInfraStructurePolicy> list =
instance.userAPIGetAllServiceContainerVirtualInfraPolicies();
  for (Iterator iterator = list.iterator(); iterator.hasNext();) {
  APICContainerVirtualInfraStructurePolicy apicContainerVirtualInfraStructurePolicy =
(APICContainerVirtualInfraStructurePolicy) iterator
   .next();
   System.out.println(" Profile Name :
"+apicContainerVirtualInfraStructurePolicy.getPolicyName());
   System.out.println(" Profile Description :
"+apicContainerVirtualInfraStructurePolicy.getPolicyDescription());
   System.out.println(" Application Profile Name :
"+apicContainerVirtualInfraStructurePolicy.getApplicationProfileName());
  }
}
```

Results

The API returns the list of APIC virtual infrastructure policies in Cisco UCS Director.

Implementation

No implementation required.

See Also

Creating an APIC Virtual Infrastructure Policy

Retrieving an APIC Virtual Infrastructure Policy

Modifying an APIC Virtual Infrastructure Policy

Deleting an APIC Virtual Infrastructure Policy

Retrieving an APIC Virtual Infrastructure Policy

Objective

Retrieve the details of a virtual infrastructure policy by passing the policy name.

Context

To view the set policy details.

Prerequisites

The virtual infrastructure policy must exist.

REST URL

```
/app/api/rest?formatType=json&opName=
apic:userAPIGetServiceContainerVirtualInfraPolicy&opData={param0:"testPolicy"}
```

Components

- policyName—The name of the virtual infrastructure policy that you want to delete.

Code

```
public class GetAPICContainerVirtualInfraPolicyExample
{
 public static void main(String[] args) throws Exception {
 CuicServer server = CuicServer.getAPI("172.22.234.33","EF0FE312B5444E7B8DE4836DDC55F4A0",
 "https", 443);
 UserAPIAPICContainer instance = new UserAPIAPICContainer(server);
  List<APICContainerVirtualInfraStructurePolicy> list =
instance.userAPIGetServiceContainerVirtualInfraPolicy("apicPolicy");
  for (Iterator iterator = list.iterator(); iterator.hasNext();) {
   APICContainerVirtualInfraStructurePolicy apicContainerVirtualInfraStructurePolicy =
 (APICContainerVirtualInfraStructurePolicy) iterator
     .next();
   System.out.println(" Profile Name :
"+apicContainerVirtualInfraStructurePolicy.getPolicyName());
   System.out.println(" Profile Description :
"+apicContainerVirtualInfraStructurePolicy.getPolicyDescription());
   System.out.println(" Application Profile Name :
"+apicContainerVirtualInfraStructurePolicy.getApplicationProfileName());
   }
}
```

Results

If the policy details are retrieved successfully, the result is true.

Implementation

Call the userAPIGetServiceContainerVirtualInfraPolicy API and pass the virtual infrastructure policy name.

See Also

Creating an APIC Virtual Infrastructure Policy

Listing All APIC Virtual Infrastructure Policy

Modifying an APIC Virtual Infrastructure Policy

Deleting an APIC Virtual Infrastructure Policy

Modifying an APIC Virtual Infrastructure Policy

Objective

Update the virtual infrastructure policy of an APIC container with the given details. The policy name cannot be edited.

Context

To update the policy of virtual infrastructure defined for an APIC container. The updated policy is used for creating a new service container template.

Prerequisites

The APIC virtual infrastructure policy must exist.

REST URL

```
/app/api/rest?formatType=json&opName=apic:
userAPIUpdateServiceContainerVirtualInfraPolicy&opData={param0"policyName":
"testPolicy","policyDescription":"updated testPolicy ","containerType":"APIC",
"applicationProfileName":"appprofile"}}
```

Response on successful execution of the API

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
"opName":"apic:userAPIUpdateServiceContainerVirtualInfraPolicy" }
```

Response on unsuccessful execution of the API

If the application profile specified in the REST URL does not exist, the API throws the following exception which suggests using a valid application profile name.

```
{ "serviceResponse":null, "serviceError":"REMOTE_SERVICE_EXCEPTION:
Application profile ghfgh does not exist, Please provide available
 application profile name", "serviceName":"InfraMgr",
"opName":"apic:userAPIUpdateServiceContainerVirtualInfraPolicy" }
```

Components

- policyName—The name of the virtual infrastructure policy.
- containerType—The type of the container must be APIC.
- applicationProfileName—The name of the application profile that you want to use for creating the container.

Code

```
public class APICContainerVirtualInfraPolicyUpdateExample
{
 public static void main(String[] args) throws Exception {
 CuicServer server = CuicServer.getAPI("172.22.234.33","EF0FE312B5444E7B8DE4836DDC55F4A0",
 "https", 443);
 UserAPIAPICContainer instance = new UserAPIAPICContainer(server);
 APICContainerVirtualInfraStructurePolicy policy = new
APICContainerVirtualInfraStructurePolicy();
 policy.setPolicyName("VIPolicy1");
 policy.setPolicyDescription("VIPolicy1 Description");
 policy.setApplicationProfileName("applicationProfileName");
   /**
   * Update the service Container Virtual Infra Policy
   */
  boolean obj = instance.userAPIUpdateServiceContainerVirtualInfraPolicy(policy);
  System.out.println(obj);
}
}
```

Results

If the virtual infrastructure policy is updated successfully, the result is true.

Implementation

Call the userAPIUpdateServiceContainerVirtualInfraPolicy API, and pass the virtual infrastructure policy name along with the necessary details that you want to be updated.

See Also

Creating an APIC Virtual Infrastructure Policy

Listing All APIC Virtual Infrastructure Policy

Retrieving an APIC Virtual Infrastructure Policy

Deleting an APIC Virtual Infrastructure Policy

Deleting an APIC Virtual Infrastructure Policy

Objective

Delete an APIC virtual infrastructure policy from Cisco UCS Director.

Context

A user belonging to the group to which the virtual account is mapped, can delete the virtual infrastructure policy.

Prerequisites

The virtual infrastructure policy must exist.

REST URL

```
/app/api/rest?formatType=json&opName=
apic:userAPIDeleteServiceContainerVirtualInfraPolicy&opData={param0:
{"policyName":"apicPolicy”}
```

Components

- policyName—The name of the virtual infrastructure policy that you want to delete.

Code

```
public class APICContainerVirtualInfraPolicyDeleteExample
{
 public static void main(String[] args) throws Exception {
 CuicServer server = CuicServer.getAPI("172.22.234.33","EF0FE312B5444E7B8DE4836DDC55F4A0",
 "https", 443);
 CuicServer server = CuicServer.getAPI("<IP address>", "<REST Key>", "https", 443);
 UserAPIAPICContainer instance = new UserAPIAPICContainer(server);
 APICContainerVirtualInfraStructurePolicy policy = new
APICContainerVirtualInfraStructurePolicy();
 policy.setPolicyName("apicPolicy");

 boolean isDeleted = instance.userAPIDeleteServiceContainerVirtualInfraPolicy(policy);

 System.out.println("is Policy Deleted ?"+isDeleted);
}
}
```

Results

If the APIC virtual infrastructure policy is deleted successfully, the result is true.

Implementation

Call the userAPIDeleteServiceContainerVirtualInfraPolicy API and pass the virtual infrastructure policy name.

See Also

Creating an APIC Virtual Infrastructure Policy

Listing All APIC Virtual Infrastructure Policy

Retrieving an APIC Virtual Infrastructure Policy

Modifying an APIC Virtual Infrastructure Policy

Managing Service Containers

Creating a Service Container with Template

Objective

Create a service container using a template.

Context

Create a container using a template to provision a VDC or VM.

Prerequisites

- The logged-in user must have permission to create a service container.
- The container template must be available in Cisco UCS Director.

REST URL

```
/app/api/rest?formatType=json&opName=fenced:userAPICreateServiceContainerWithoutCatalog&opData=
{param0:"SDKCont",param1:2,param2:"SDKContTemplate"}
```

Components

The container template is required.

Code

```
public class CreateServiceContainerExample
{
 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 srId = instance.userAPICreateServiceContainerWithoutCatalog("SDKCont", 2,
"SDKContTemplate");
 System.out.println(" Service Container Request id "+srId);
 }
}
```

Results

The service request ID for creating service container is returned.

Implementation

Call the userAPICreateServiceContainerWithoutCatalog API and pass the container name, group ID, and container template to create a service container.

See Also

Retrieving a Service Container

Retrieving a Service Container with Catalog

Listing all Service Containers in Cisco UCS Director

Adding a Tier to a Container VM

Adding a Tier to an APIC Container

Adding a Virtual Network Interface Card to a Container VM

Deleting a Service Container

Retrieving a Service Container

Objective

Retrieve the details of a service container from Cisco UCS Director.

Context

Identify the container availability to provision a VDC or VM.

Prerequisites

The ID of the service container must be known.

REST URL

```
/app/api/rest?formatType=json&opName=fenced:userAPIGetServiceContainerData&opData={param0:2}
```

Components

Service container id—The ID of the service container for which you need to view the details.

Code

```
public class GetServiceContainerDataExample
{
 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 containerDataObject = instance.userAPIGetServiceContainerData(2);

   System.out.println("Container Id = "+containerDataObject.getContainerId());
  System.out.println("Gateway  = "+containerDataObject.getVmRequestBundle().getGateway());

 }
      }
```

Results

The details of the service container with the specific ID are returned.

Implementation

Call the userAPIGetServiceContainerData API by passing the service container ID to retrieve the service container details.

See Also

Creating a Service Container with Template

Retrieving a Service Container with Catalog

Listing all Service Containers in Cisco UCS Director

Adding a Tier to a Container VM

Adding a Tier to an APIC Container

Adding a Virtual Network Interface Card to a Container VM

Deleting a Service Container

Retrieving a Service Container with Catalog

Objective

Retrieve the details of a service container along with the catalog item used for creating the service container.

Context

Identify the container availability to provision a VDC or VM.

Prerequisites

The logged-in user must have permission to retrieve the service container details.

REST URL

```
/app/api/rest?formatType=json&opName=fenced:userAPIGetServiceContainerDetails&opData={param0:2}
```

Components

container id—The ID of the service container for which you need to view the details.

Code

```
public class GetServiceContainerDetails
{
 public static void main(String[] args) throws Exception
 {
  CuicAPIClient client = new CuicAPIClient("10.23.210.119", 80,
"38B101A62AF14024964D6A87C23C09DE", "http");
  List listObj = new ArrayList();
  //Paas the container id
  listObj.add("4");
  JsonElement jsonResponse =
client.sendJSONRequest("fenced:userAPIGetServiceContainerDetails", listObj);
  JsonArray array = jsonResponse.getAsJsonObject().get("rows").getAsJsonArray();
  for (int count = 0; count < array.size(); count++)
   {
   JsonElement jsonElement = array.get(count);
   JsonObject jsonObject = jsonElement.getAsJsonObject();
   System.out.println("Overview_Group:  " +
jsonObject.get("Overview_Group").getAsString());
   System.out.println("Overview_ID:  " + jsonObject.get("Overview_ID").getAsString());
   System.out.println("Overview_containerName:  " +
jsonObject.get("Overview_containerName").getAsString());
   System.out.println("Overview_VM_Counts:  " +
jsonObject.get("Overview_VM_Counts").getAsString());
   System.out.println("Overview_ServiceRequestStatus:  "
     + jsonObject.get("Overview_ServiceRequestStatus").getAsString());
   System.out.println("vInfraPolicyInfo_ContainerType:  "
     + jsonObject.get("vInfraPolicyInfo_ContainerType").getAsString());
   System.out.println("Policy_VirtualCompute:  " +
jsonObject.get("Policy_VirtualCompute").getAsString());
   System.out.println("Policy_VirtualNetwork:  " +
jsonObject.get("Policy_VirtualNetwork").getAsString());
   System.out.println("Policy_VirtualSystem:  " +
jsonObject.get("Policy_VirtualSystem").getAsString());
   System.out.println("Policy_VirtualStorage:  " +
jsonObject.get("Policy_VirtualStorage").getAsString());
   }
  }
}
```

Results

The following service container details are returned:

- Overview_Group: Default Group
- Overview_ID: 4
- Overview_containerName: cont1
- Overview_VM_Counts: Container is Empty
- Overview_ServiceRequestStatus: Complete
- vInfraPolicyInfo_ContainerType: Fenced Virtual
- Policy_VirtualCompute: VNX_CLOUD69 - Default Computing Policy
- Policy_VirtualNetwork: VNX_CLOUD69 - Default Network Policy
- Policy_VirtualSystem: Default System Policy
- Policy_VirtualStorage: VNX_CLOUD69 - Default Storage Policy

Implementation

Call the userAPIGetServiceContainerDetails API by passing the container ID to retrieve the service container details.

See Also

Creating a Service Container with Template

Retrieving a Service Container

Listing all Service Containers in Cisco UCS Director

Adding a Tier to a Container VM

Adding a Tier to an APIC Container

Adding a Virtual Network Interface Card to a Container VM

Deleting a Service Container

Listing all Service Containers in Cisco UCS Director

Objective

Retrieve and display all the service containers that are available in Cisco UCS Director.

Context

To view the list of service containers (APIC and fenced) in Cisco UCS Director.

Prerequisites

The service containers must be available in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=apic:userAPIGetAllServiceContainers&opData={}
```

Response

```
{ "serviceResult":[{"id":2,"containerType":"APIC","containerName":"cdevBld",
"containerLabel":"","containerState":2,"tenantIdentity":"VNX_APIC185@cdev23",
"privateNetwork":null,"hostsNumberPerTier":"","groupId":4,"vdcId":2,"provisionedTime":
1464001006693,"termiantionTime":-1,"srId":1107,"enableDR":false,"primaryServiceContainerId":
-1,"serviceContainerRole":"PRIMARY","configureResourceLimit":true,"cpu":1.0,"ncpu":3,
"memory":3.0,"provisionedDiskGBLimit":20.0,"halfWidthPhysicalServerLimit":1,
"fullWidthPhysicalServerLimit":1,"enableNetworkMgmt":true,"networkThrougput":"100M",
"customTierLabels":null,"vmLabels":{"list":[],"moTypeName":"com.cloupia.model.
serviceContainer.VMsLabelCustomizationConfig","validatorName":null},
"customTierProperty":null}], "serviceError":null, "serviceName":"InfraMgr",
"opName":"apic:userAPIGetAllServiceContainers" }
```

Components

None

Code

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Properties;

import com.cisco.cuic.api.client.APITabularReport;
import com.cisco.cuic.api.client.ContainerVirtualMachine;
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.client.JSON;
import com.cisco.cuic.api.models.UserAPIFencedContainer;
import com.cisco.cuic.api.models.UserAPIGlobal;
import com.cisco.cuic.api.models.servicecontainer.APIFencedVirtualContainerTemplateConfig;
import com.cisco.cuic.api.models.servicecontainer.APIFencedVirtualMachineConfig;
import com.cisco.cuic.api.models.servicecontainer.APIServiceContainerTemplate;
import com.cisco.cuic.api.models.servicecontainer.ContainerDataObjects;
import com.cisco.cuic.api.models.servicecontainer.DFAConfig;
import com.cisco.cuic.api.models.servicecontainer.FencedContainerOptions;
import com.cisco.cuic.api.models.servicecontainer.FencedContainerPolicies;
import com.cisco.cuic.api.models.servicecontainer.FencedContainerWorkflowConfig;
import com.cisco.cuic.api.models.servicecontainer.FencedNetwork;
import com.cisco.cuic.api.models.servicecontainer.FencedVirtualMachineNetworkConfig;
import com.cisco.cuic.api.models.servicecontainer.NetworkConfigProvisionedPortGroup;
import com.cisco.cuic.api.models.servicecontainer.NetworkConfigRequest;
import com.cisco.cuic.api.models.servicecontainer.NetworkConfigResult;
import com.cisco.cuic.api.models.servicecontainer.NetworkTopology;
import com.cisco.cuic.api.models.servicecontainer.OutboundACL;
import com.cisco.cuic.api.models.servicecontainer.OwnerInfo;
import com.cisco.cuic.api.models.servicecontainer.PortMapping;
import com.cisco.cuic.api.models.servicecontainer.ServiceContainer;
import com.cisco.cuic.api.models.servicecontainer.ServiceContainervInfraPolciyDef;
import com.cisco.cuic.api.models.servicecontainer.ServiceContainervInfraPolicy;
import com.cisco.cuic.api.models.servicecontainer.VirtualSwitch;
import com.cloupia.sdk.api.CloupiaClient;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

public class TestApplicationContainer
{
 private static final String REST_SERVER_PROPERTIES = "rest-server.properties";
 private static final String UCSM_CONTEXT_NAME = "ucsm";
 private Properties props;
 private CloupiaClient client;
 private Map<String, String> reportLabelIdMap;
 private JsonObject obj;
 private JsonParser parser;

 public TestApplicationContainer()
{
   }

 public static void main(String[] args) throws Exception {
  TestApplicationContainer obj = new TestApplicationContainer();
  CuicServer server = CuicServer.getAPI("172.22.234.243", "
CF87FA987C8F4BBF814F2BB68CA6A823", "https", 443);
  UserAPIFencedContainer fencedC = new UserAPIFencedContainer(server);
  executeGetServiceContainerDetails(fencedC);
 }
 public static void executeGetServiceContainerDetails(UserAPIFencedContainer fencedC)
throws Exception
{
  APITabularReport atr = new APITabularReport();
  atr = fencedC.userAPIGetServiceContainerDetails(1);

  List<Map<String, Object>> listofMaps = atr.getRows();
  // System.out.println("rows " + atr.getRowCount());
  for (Map<String, Object> map : listofMaps) {
   for (Entry<String, Object> entry : map.entrySet()) {
    // System.out.println(entry.getKey() + ":");
    String o = entry.toString();
    System.out.println(o);
   }
  }
 }
```

Results

The code returns a list of service containers in Cisco UCS Director.

Implementation

No implementation required.

See Also

Creating a Service Container with Template

Retrieving a Service Container

Retrieving a Service Container with Catalog

Adding a Tier to a Container VM

Objective

Add a tier to a VM in an APIC container.

Note From Cisco UCS Director Release 6.x and later releases, the userAPIAddTierToContainerVM API is deprecated. You can use the userAPIAddTierToContainer API to add a tier to a container. For more details, see Adding a Tier to an APIC Container , on page 47.

Context

To add a tier to an APIC container VM.

Prerequisites

- The APIC container must be available in Cisco UCS Director.
- The VM must be available in the APIC container.

REST URL

Request

```
/app/api/rest?formatType=json&opName=
apic:userAPIAddTierToContainerVM&opData={param0:{"containerId":2,"tierName":"app",
"tierLabel":"app","isIsolated":false,"parentTierName":""}}
```

Response

```
{ "serviceResult":2197, "serviceError":null, "serviceName":"InfraMgr",
"opName":"apic:userAPIAddTierToContainerVM" }
```

Components

The parameters of the userAPIAddTierToContainerVM API are:

- int containerId—The ID of the APIC container.
- String tierName—The name of the tier that you want to add to the APIC container.
- String tierLabel—The label of the tier.
- boolean isIsolated—Set to true to associate the tier with the parent tier in the APIC container.
- String parentTierName—The parent tier name to which the tier needs to be associated with. Provide the parent tier name when the isIsolated parameter is set to true.

Code

```
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.apic.APIAPICAddTierToContainerParams;
import com.cisco.cuic.api.models.apic.APIAPICAddvNICToContainerVMParams;
import com.cisco.cuic.api.models.apic.userAPIContainerandContract;

public class userAPIAddTierToContainerVMTest
{
  public static void main(String[] args)throws Exception {
  CuicServer api = CuicServer.getAPI("172.22.234.243", "CF87FA987C8F4BBF814F2BB68CA6A823",
 "http", 80);
  userAPIContainerandContract instance = new userAPIContainerandContract(api);
  APIAPICAddTierToContainerParams param=new APIAPICAddTierToContainerParams();
  param.setContainerId(2);
  param.setIsolated(false);
  param.setParentTierName("");
  param.setTierLabel("apps");
  param.setTierName("apps");
  int requestId=instance.userAPIAddTierToContainerVM(param);
  System.out.println(requestId);
 }
}
```

Results

Returns a service request ID on successful addition of tier to the APIC container VM.

Implementation

Use the userAPIAddTierToContainerVM API to add a tier to the APIC container VM.

See Also

Creating a Service Container with Template

Retrieving a Service Container

Retrieving a Service Container with Catalog

Listing all Service Containers in Cisco UCS Director

Adding a Tier to an APIC Container

Adding a Virtual Network Interface Card to a Container VM

Deleting a Service Container

Adding a Tier to an APIC Container

Objective

Add a tier to an APIC container.

Context

To add a tier to an APIC container. As the userAPIAddTierToContainerVM API is deprecated, we recommend you to use the userAPIAddTierToContainer API.

Prerequisites

The APIC container must be available in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=apic:userAPIAddTierToContainer&opData=
{param0:{"containerId":1,"tierName":"app","tierLabel":"app","isIsolated":false,
"parentTierName":""}}
```

Response

```
{ "serviceResult":17, "serviceError":null, "serviceName":"InfraMgr",
"opName":"apic:userAPIAddTierToContainer" }
```

Components

- Int containerId—The ID of the APIC container.
- String tierName—The name of the tier that you want to add to the APIC container.
- String tierLabel—The Label of the tier.
- boolean isIsolated—Set to true to associate the tier with the parent tier in the APIC container.
- String parentTierName—The parent tier name to which the tier needs to be associated with. Provide the parent tier name when the isIsolated parameter is set to true.

Code

```
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.UserAPIAPICContainer;
import com.cisco.cuic.api.models.apic.APIAPICAddTierToContainerParams;

public class TestUserAPIAddTierToContainer {

 public static void main(String[] args) {
  CuicServer server = CuicServer.getAPI("172.22.234.172",
"91E7A134048F45FD88A29FCBBFD2C233", "https", 443, 12000, "none", "");
   UserAPIAPICContainer instance = new UserAPIAPICContainer(server);
   APIAPICAddTierToContainerParams apicAddTierToContainerParams = new
APIAPICAddTierToContainerParams();
   apicAddTierToContainerParams.setContainerId(1);
   apicAddTierToContainerParams.setTierName("app");
   apicAddTierToContainerParams.setTierLabel("app");
   apicAddTierToContainerParams.setIsolated(false);
   apicAddTierToContainerParams.setParentTierName("");
   try {
   int requestId = instance.userAPIAddTierToContainer(apicAddTierToContainerParams);
   System.out.println("Service Request Id = "+requestId);
  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }
 }

}
```

Results

Returns a service request ID on successful addition of tier to the APIC container.

Implementation

Call the userAPIAddvNICToContainerVM API to add a vNIC to an APIC container VM.

See Also

Creating a Service Container with Template

Retrieving a Service Container

Retrieving a Service Container with Catalog

Listing all Service Containers in Cisco UCS Director

Adding a Virtual Network Interface Card to a Container VM

Objective

Add a virtual Network Interface Card (vNIC) to a VM in an APIC container.

Context

To add a vNIC to an APIC container VM.

Prerequisites

- The APIC container must be available in Cisco UCS Director.
- The VM must be available in the APIC container.

REST URL

Request

```
/app/api/rest?formatType=json&opName=
apic:userAPIAddvNICToContainerVM&opData={param0:{"containerId":"2","vmId":"1",
"vmUsername":"root","vmPassword":"cloupia123","networkName":"Con"}}
```

Response

```
{ "serviceResult":2190, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"apic:userAPIAddvNICToContainerVM" }
```

Components

The parameters of the userAPIAddvNICToContainerVM API are:

- int containerId—The ID of the APIC container.
- int vmId—The ID of the VM to which you need to add vNIC.
- String vmUsername—The username that is used to access the VM.
- String vmPassword—The password that is used to access the VM.
- String networkName—The tier or network within the same application container where the VM resides in.

Code

```
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.UserAPIVMware;
import com.cisco.cuic.api.models.apic.APIAPICAddvNICToContainerVMParams;
import com.cisco.cuic.api.models.apic.userAPIContainerandContract;

public class userAPIAddvNICToContainerVMTest
{
  public static void main(String[] args)throws Exception {
  CuicServer api = CuicServer.getAPI("172.22.234.243", "CF87FA987C8F4BBF814F2BB68CA6A823",
 "http", 80);
  userAPIContainerandContract instance = new userAPIContainerandContract(api);
  APIAPICAddvNICToContainerVMParams param=new APIAPICAddvNICToContainerVMParams();
  param.setContainerId("2");
  param.setNetworkName("Con");
  param.setVmId("1");
  param.setVmUsername("root");
  param.setVmPassword("cloupia123");
  int requestId=instance.userAPIAddvNICToContainerVM(param);
  System.out.println(requestId);
 }
}
```

Results

Returns a service request ID on successful addition of vNIC to the APIC container VM.

Implementation

Use the userAPIAddvNICToContainerVM API to add a vNIC to an APIC container VM.

See Also

Creating a Service Container with Template

Retrieving a Service Container

Retrieving a Service Container with Catalog

Listing all Service Containers in Cisco UCS Director

Adding a Tier to a Container VM

Adding a Tier to an APIC Container

Deleting a Service Container

Deleting a Service Container

Objective

Delete a service container from Cisco UCS Director.

Context

Reuse the resources that are occupied by the provisioned VM in a container. After the usage of VM, the VM provisioning is rolled back. Then, the container must be deleted to release the resources allocated for the VM.

Prerequisites

- The logged-in user must have permission to delete a service container.
- The container must not have any VMs.

REST URL

```
/app/api/rest?formatType=json&opName=fenced:userAPIDeleteServiceContainer&opData={param0:3}
```

Components

containerId—The ID of the service container to be deleted.

Code

```
public class DeleteServiceContainerDataExample
{
 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 srId = instance.userAPIDeleteServiceContainer(2);
 System.out.println("Delete SR id  "+srId);
 }
}
```

Results

The service container is deleted and a service request ID is returned.

Implementation

Delete a service container by passing the service container ID.

See Also

Creating a Service Container with Template

Retrieving a Service Container

Retrieving a Service Container with Catalog

Listing all Service Containers in Cisco UCS Director

Adding a Tier to a Container VM

Adding a Tier to an APIC Container

Adding a Virtual Network Interface Card to a Container VM

Managing Contracts

Creating a Contract

Objective

Create a contract between two networks in an APIC container.

Context

To create a contract between two networks.

Prerequisites

APIC container must be present in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=
apic:userAPICreateContract&opData={param0:{"containerId":2,"ruleName":"rule1",
"ruleDescription":"sample","sourceNetwork":"web","destNetwork":"app",
"createRule":true,"protocol":"TCP","applyBothDirections":true,"sourcePortStart":"10",
"sourcePortend":"20","destinationPortStart":"30","destinationPortEnd":"40",
"fireallAction":1000,"enableStatefull":true}}
```

Response

```
{ "serviceResult":2166, "serviceError":null, "serviceName":"InfraMgr",
"opName":"apic:userAPICreateContract" }
```

Components

The parameters of the userAPICreateContract API are:

- int containerId—The unique ID of the APIC container for which the contract is created.
- String ruleName—The name of the rule. This rule is used internally by Cisco UCS Director for its own reference.
- String ruleDescription—The description of the rule. The rule filter name is created on APIC and is autogenerated using the container name.
- String sourceNetwork—The source network to which you want to apply the contract rule.
- String destNetwork—The destination network to which you want to apply the contract rule.
- boolean createRule—Set to true to create a rule.
- String protocol—The protocol for communication.
- boolean applyBothDirections—Set to true to apply the same contract for traffic from source to destination.
- int sourcePortStart—The starting port number of the specified port range of source.
- int sourcePortend—The ending port number of the specified port range of source.
- int destinationPortStart—The starting port number of the specified port range of destination.
- int destinationPortEnd—The ending port number of the specified port range of destination.
- int fireallAction—Leave this parameter empty.
- boolean enableStatefull—Set to true to enable stateful.

Code

```
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.apic.APIAPICAddTierToContainerParams;
import com.cisco.cuic.api.models.apic.APICCreateContractParams;
import com.cisco.cuic.api.models.apic.userAPIContainerandContract;

public class userAPICreateContractTest
{
 public static void main(String[] args)throws Exception {
  CuicServer api = CuicServer.getAPI("172.22.234.243", "CF87FA987C8F4BBF814F2BB68CA6A823",
 "http", 80);
  userAPIContainerandContract instance = new userAPIContainerandContract(api);
  APICCreateContractParams param=new APICCreateContractParams();
  param.setContainerId(2);
  param.setApplyBothDirections(true);
  param.setCreateRule(true);
  param.setDestinationPortEnd("40");
  param.setDestinationPortStart("30");
  param.setDestNetwork("app");
  param.setSourceNetwork("web");
  param.setEnableStatefull(true);
  param.setFireallAction(100);
  param.setProtocol("TCP");
  param.setRuleDescription("sdk");
  param.setRuleName("Rule123");
  param.setSourcePortend("20");
  param.setSourcePortStart("10");
  int requestId=instance.userAPICreateContract(param);
  System.out.println(requestId);
 }
}
```

Results

A unique contract ID is returned after successful creation of a contract in the Cisco UCS Director server.

Implementation

Use the userAPICreateContract API and pass the necessary data to create a contract.

See Also

Deleting a Contract

Deleting a Contract

Objective

Delete a contract between networks in an APIC container.

Context

The catalog can be deleted by a system administrator.

Prerequisites

APIC container must be present in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=
apic:userAPIDeleteContract&opData={param0:{"ruleId":"rule1","srcNetwork":"web",
"destNetwork":"app"}}
```

Response

```
{"serviceResult":2176, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"apic:userAPIDeleteContract" }
```

Components

The parameters of the userAPIDeleteContract API are:

- String ruleName—The name of the rule that you want to delete.
- String srcNetwork—The source network from which you want to delete the contract rule.
- String destNetwork—The destination network from which you want to delete the contract rule.

Code

```
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.apic.APICCreateContractParams;
import com.cisco.cuic.api.models.apic.APICDeleteContractParams;
import com.cisco.cuic.api.models.apic.userAPIContainerandContract;

public class userAPIDeleteContractTest
{
  public static void main(String[] args)throws Exception {
  CuicServer api = CuicServer.getAPI("172.22.234.243", "CF87FA987C8F4BBF814F2BB68CA6A823",
 "http", 80);
  userAPIContainerandContract instance = new userAPIContainerandContract(api);
  APICDeleteContractParams param=new APICDeleteContractParams();
  param.setDestNetwork("app");
  param.setRuleId("Rule123");
  param.setSrcNetwork("web");
  int requestId=instance.userAPIDeleteContract(param);
  System.out.println(requestId);
 }
}
```

Results

Returns the service request ID on successful deletion of the contract between the source and destination networks.

Implementation

Call the userAPIDeleteContract API and pass the rule name, source network, and destination network to delete the contract between the source and destination networks.

See Also

Creating a Contract

Managing Virtual Machines

Provisioning a VM

Objective

You can provision a VM on VMware using a standard catalog.

Context

To provision a VM on VMware.

Prerequisites

Ensure that the following are available in Cisco UCS Director:

- VM cloud
- VMware system
- Compute, network, and storage policies
- VDC
- Standard catalog

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPIProvisionRequest&opData=
{param0:{"catalogName":"catalog_1","vdcName":"vdc_1","userID":"admin","selectedDiskDataStore":
```

```
"[{'diskName':'Hard disk 1','diskSize':'10','datastore':'QA_DS01','diskType':'0'},
{'diskName':'Hard disk 2','diskSize':'10','datastore':'QA_DS01','diskType':'0'}]"}}
```

Response

```
{ "serviceResult":52, "serviceError":null, "serviceName":"InfraMgr",
"opName":"userAPIProvisionRequest" }
```

Components

The following parameters are mandatory for provisioning a VM:

- String catalogName—The catalog name from which the VM is provisioned.
- String vdcName—The name of the VDC.
- String userID—The unique ID of the user.

Note When you use catalogName, vdcName, and userId, a VM is created based on various policies (VMware system policy, VMware compute policy, VMware storage policy, and VMware network policy) defined in Cisco UCS Director.

The following parameters are optional for provisioning a VM:

- int durationHours—The amount of time that the VM is active.
- int beginTime—The start time at which the VM is provisioned.
- int quantity—The number of VMs to be provisioned.
- int memoryMB—The primary memory for the VM being provisioned, in GB.
- int diskGB—The hard disk for the VM provisioning in GB.
- int cores—The number of vCPU for the VM being provisioned.
- int estimatedCost—The estimated cost for the service request.
- String comments—Any comments for VM provisioning.
- String additionalInfo—Any additional information for VM provisioning.
- int chargeFrequency—The frequency at which the user has to be charged. This parameter accepts the following values: Hourly or Monthly.
- String nicAliasName—The name of network interface card (NIC) alias.
- String nicPortGroupName—The port group name of the NIC.

  Note The format of the NIC port group name is as follows: cloudName@SwitchName@portgroup type@Network name. For example, "nicPortGroupName":"vmware_cloud_82@vSwitch0@Virtual Machine Portgroup@VM Network".

- boolean resourceAllocated—Set as true to allocate resource for the VM.

- String allocatedHost—The host name allocated for the VM.
- String allocatedDataStore—The datastore allocated for the VM.
- String allocatedResourcePool—The resource allocated for the VM.
- String altAllocatedHost—The alternate host name allocated for the VM.
- String altAllocatedDataStore—The alternate datastore allocated for the VM.
- String altAllocatedResourcePool—The alternate resource pool allocated for the VM.
- int customStartupMemory—The custom startup memory size set for the VM.
- int customMaxMemory—The maximum memory size that can be allotted for the VM.
- int customMemoryBuffer—The custom memory buffer for the VM.
- boolean customMemoryConfig—Set as true to allow memory configuration.
- String customStoragePolicy—The storage policy for provisioning a VM with respect to the VDC. If the storage policy is defined in the customStoragePolicy parameter, the UserAPIProvisionRequest API refers the storage policy for provisioning a VM. If the storage policy is not defined in the customStoragePolicy parameter, the UserAPIProvisionRequest API refers the default storage policy of the specified VDC for provisioning a VM.
- String allocatedCluster—The cluster allocated for the VM.
- int customCpuSockets—The number of CPU sockets allotted for the VM.
- int customCpuCoresPerSocket—The number of CPUs allotted per socket.
- String altAllocatedCluster—The alternate cluster allocated for the VM.
- String allocatedAddnlDatastores—The additional datastore allocated for the VM.
- String altAllocatedAddnlDatastores—The alternate additional datastore allocated for the VM.
- String altAllocatedAddnlVNICs—The alternate additional vNICs allocated for the VM.
- String altAllocatedAddnlVNICsIpv6—The alternate additional vNICs that support IPv6, allocated for the VM.
- String selectedDiskDataStore—The data store that is chosen for hard disk defined in the VMware storage policy. The hard disk name must be in the format: `{'diskName':'`*`diskName`*`','diskSize':'`*`diskSizeinGB`*`','datastore':'`*`datastoreName`*`','diskType':'`*`diskType`*`'}`. The valid disk types are:
  - 0—System
  - 1—Data
  - 2—Database
  - 3—Log
  - 4—Swap

Example for defining two hard disks: `"selectedDiskDataStore":"[{'diskName':'Hard disk 1','diskSize':'5','datastore':'esxi02_boot','diskType':'0'},{'diskName':'Hard disk 2','diskSize':'5','datastore':'esxi02_boot','diskType':'0'}]"`

In this example, all disks are provisioned on single data store. If you want to provision multiple disks on different data stores, uncheck the **Provision all disks in single datastore** check box in the policy catalog.

- String actionId—The workflow actionId that is used for the VM provisioning.
- String vmName—The name of the VM being provisioned.
- String vdcCategory—The category of the VDC.
- String windowsLicensePool—The pool of windows licenses.
- String templateUserId—The user ID of the template.
- String templatePassword—The password of the template.
- String credentialOption—The VM access credentials.
- boolean provisionAllDisk—Set to true to provision all disks in a single datastore.
- boolean enableGuestCustomization—Set to true to enable guest customization in the VM.
- boolean enablePostProvisioningCustomActions—Set to true to enable an orchestration workflow that is executed after VM provisioning.
- String workflow—The workflow for VM provisioning.
- Int vmId—The identity of the VM.
- Int vMAppChargeFrequency—The frequency at which the VM application has to be charged. This parameter accepts the following values: Hourly or Monthly.
- Int activeVMAppCost—The cost for the application that is included in the template.
- Int inactiveVMAppCost—The cost to this catalog of a VM in inactive state per hour or month.
- boolean useLinkedClone—Set to true to use the linked clone feature.
- Int snapshotId—The snapshot ID for the VM.
- String snapshotKey—The key of the snapshot.
- String newSnapshotName—The name of the new snapshot for the VM.
- boolean isHighlyAvailable—Set to true if high availability support is available in VM.
- List postProvWFUserInputs—Set the list of user input for workflow that you want to execute post provisioning of VM.

Code

```
import com.cisco.cuic.api.models.UserAPIGlobal;
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.client.APIProvisionParams;
public class TestuserAPIProvisionRequest {
public static void main(String[] args) {
  CuicServer server = CuicServer.getAPI("172.29.110.222",
"6F0063A7F7654561A790EACCE1E8626F", "https", 443);
```

```
    UserAPIGlobal instance = new UserAPIGlobal(server);
    APIProvisionParams params=new APIProvisionParams();
    int srId;
    params.setCatalogName("catalog_1");
    params.setVdcName("vdc_1");
    params.setUserID("admin");
    String selectedDiskDataStore="[{'diskName':'Hard disk
  1','diskSize':'5','datastore':'QA_DS01','diskType':'0'},
  {'diskName':'Hard disk 2','diskSize':'-1','datastore':'QA_DS01','diskType':'0'}]";
    params.setSelectedDiskDataStore(selectedDiskDataStore); //Optional
     try {
     srId=instance.userAPIProvisionRequest(params);
     System.out.println("Service Request Id for VM provision request = "+srId);
    } catch (Exception e) {

     e.printStackTrace();
    }
   }
  }
```

Results

The service request ID for provisioning a VM is returned.

Implementation

Call the userAPIProvisionRequest API and pass the required parameters to provision a VM.

See Also

Powering On a VM

Rebooting a VM

Adding a Virtual Network Interface Card to a VM

Powering Off a VM

Powering On a VM

Objective

Power on a VM.

Context

Manage the VMs that are available on VDC.

Prerequisites

VM must be available for access.

REST URL

```
/app/api/rest?formatType=json&opName=userAPIExecuteVMAction&opData={param0:5,
param1:"powerOn",param2:"Power On sample test"}
```

Components

- int vmId—The ID of the virtual machine that need to be powered on.
- String actionName—Set the value as powerOn.
- String comments—Optional. Additional information, if any.

Code

```
public class userAPIExecuteVMActionExample
{
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 statusMsg = instance.userAPIExecuteVMAction(1,"powerOn","Testing");
   System.out.println(" Response msg is "+statusMsg);
 }
}
```

Results

Provide a status for the powered on VM.

Implementation

Pass the VM ID and set the action name as power on in the userAPIExecuteVMAction API to power on the VM.

See Also

Provisioning a VM

Rebooting a VM

Adding a Virtual Network Interface Card to a VM

Powering Off a VM

Rebooting a VM

Objective

Reboot a VM.

Context

Manage the VMs available on VDC.

Prerequisites

VM must be available for access.

REST URL

```
/app/api/rest?formatType=json&opName=userAPIExecuteVMAction&opData={param0:5,
param1:"reboot",param2:"Reboot VM"}
```

Components

- int vmId—The ID of the virtual machine that need to be reboot.
- String actionName—Set the value as reboot.
- String comments—Optional. Additional information, if any.

Code

```
public class userAPIExecuteVMActionExample {
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 statusMsg = instance.userAPIExecuteVMAction(1, "Reboot", "Testing");
 System.out.println(" Response msg is "+statusMsg);}
}
```

Results

Provides a status of the reboot VM.

Implementation

Pass the VM ID and set the action name as reboot in the userAPIExecuteVMAction API to reboot the VM.

See Also

Provisioning a VM

Powering On a VM

Adding a Virtual Network Interface Card to a VM

Powering Off a VM

Adding a Virtual Network Interface Card to a VM

Objective

Add a virtual Network Interface Card (vNIC) to a VM to bridge a network. The vNIC must be available for both APIC and Fenced containers, and for standard catalog VMs.

Context

Virtualization of a network.

Prerequisites

Ensure that the portGroupIdentity, static IP pool, valid container template (APIC or fenced), and standard catalog VM are available.

REST URL

Request with DHCP set to True

```
app/api/rest?formatType=json&opName=genericvm:
userAPIAddVMNICs&opData={param0:{"vmId":531,"vNICConfig":
[{"portGroupIdentity":"vmware117@172.29.110.75@vSwitch0@VM Network@Virtual
 Machine Portgroup","isDHCP":false,"staticIpPool":"10.10.20.8","subnetMask":
"255.255.255.0","gateway":"10.10.20.1"}]}}
```

Request with DHCP set to False

```
/app/api/rest?formatType=json&opName=genericvm:
userAPIAddVMNICs&opData={param0:{"vmId":49,"vNICConfig":
[{"portGroupIdentity":"vmware117@172.29.110.75@vSwitch0@VM Network@Virtual
Machine Portgroup","isDHCP":false,"staticIpPool":"","subnetMask":"","gateway":""}]}}
```

Components

- vmId—ID of the VM to which you need to add vNIC.
- vNICConfig—The vNIC configuration to be added to VM. Need to set the following VM configuration:

- portGroupIdentity—ID of the portgroup to be assigned to vNIC. The allowed format of the port group identity is <cloudName>@<hostName>@<switchName>@<portGroupName>@<portGroupType>.
- isDHCP—Set to True to enable the DHCP configuration.
- staticIpPool—This field is optional when the isDHCP parameter is set to True. The IP address for the static IP configuration.
- subnetMask—This field is optional when the isDHCP parameter is set to true. The subnet mask address for the static IP configuration.
- gateway—This field is optional when the isDHCP parameter is set to true. The gateway address for the static IP configuration.

Code

```
public static void main(String[] args) throws Exception
{
 CuicServer api = CuicServer.getAPI("172.31.234.172", "E052D5B0D1BD4B3199DEB36620AA0004",
 "http", 80);
 UserAPIVMware instance = new UserAPIVMware(api);

 List<VMNICsInputConfig> vNICConfig = new ArrayList<VMNICsInputConfig>();
 VMNICsInputConfig nicConfig = new VMNICsInputConfig();
 nicConfig.setPortGroupIdentity("vmware169@172.31.232.176@vSwitch0@ctr-sdk-pg-lan0@Virtual
 Machine Portgroup");
 nicConfig.setDHCP(true);
 nicConfig.setStaticIpPool(null);
 nicConfig.setSubnetMask(null);
 nicConfig.setGateway(null);

 vNICConfig.add(nicConfig);

 APIVMNICInputParams param = new APIVMNICInputParams();
 param.setVmId(38);
 param.setvNICConfig(vNICConfig);
 APIVMNICOutputDetails output = instance.userAPIAddVMNICs(param);
 System.out.println("VM Id: "+output.getVmId());
   for(VMNICOutputDetails details : output.getvNicDetails()){
   System.out.println("Adaptor Name: "+details.getAdapterName());
   System.out.println("MAC Address: "+details.getMacAddress());
   System.out.println("Static IP: "+details.getStaticIpPool());
   System.out.println("Subnet Mask: "+details.getSubnetMask());
   System.out.println("Gateway: "+details.getGateway());
  }
}
```

Results

vNIC is added to the VM.

Sample Result

- VM Id: 38
- Adaptor Name: eth1
- MAC Address: 00:50:56:81:76:ba
- Static IP: null

- Subnet Mask: null
- Gateway: null

REST URL Response

```
{ "serviceResult":{"vmId":49,"vNicDetails":[{"adapterName":"eth1",
"macAddress":"00:50:56:8b:73:8b","staticIpPool":"10.10.10.0",
"subnetMask":"255.255.255.0","gateway":"10.10.10.1"}]}, "serviceError":null,
 "serviceName":"InfraMgr", "opName":"genericvm:userAPIAddVMNICs" }
```

Implementation

- For the DHCP configuration, set the isDHCP parameter as true. Once the DHCP is configured, the IP address is assigned dynamically.
- For the static configuration, set the isDHCP parameter as false and provide values for the staticIPPool, subnetMask, and gateway parameters.
- If the selected VM is associated to a container, the isDHCP parameter is set to false, and the values for staticIPPool, subnetMask, and gateway are not provided, the system checks for the container template network configuration to get the static IP configuration.

See Also

Provisioning a VM

Powering On a VM

Rebooting a VM

Powering Off a VM

Powering Off a VM

Objective

Power off a VM.

Context

Manage the VMs available on VDC.

Prerequisites

VM must be available for access.

REST URL

```
/app/api/rest?formatType=json&opName=userAPIExecuteVMAction&opData={param0:5,
param1:"powerOff",param2:"Power off sample test"}
```

Components

- int vmId—The ID of the virtual machine that need to be powered off.
- String actionName—Set the value as powerOff.
- String comments—Optional. Additional information, if any.

Code

```
public class userAPIExecuteVMActionExample
{
```

```
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 statusMsg = instance.userAPIExecuteVMAction(1,"powerOff","Testing");
  System.out.println(" Response msg is "+statusMsg);
 }
}
```

Results

Provide a status for the powered off VM.

Implementation

Pass the VM ID and set the action name as power off in the userAPIExecuteVMAction API to power off the VM.

See Also

Provisioning a VM

Powering On a VM

Rebooting a VM

Adding a Virtual Network Interface Card to a VM

Setting up a VMware VM Guest and Executing VIX Script

Objective

Set up a VMware VM guest to perform certain operations (such as, power on a VM, power off a VM, and rename a VM) on the target VM.

Context

Execute VIX script on the target VM.

Prerequisites

VMware tools need installed on the VM.

REST URL

```
/app/api/rest?formatType=json&opName=genericvm:userAPIExecuteVIXScript&opData={param0:355,
param1:"root",param2:"cloupia123",param3:"/bin/echo \"Hello\";/bin/echo 'NPROXY=\$1'
"}
```

Components

VM ID, credentials, and VIX script are required.

Code

```
public class VIXScriptExecuteUsingGuestSetup
{
 public static void main(String[] args)
 {
  CuicServer server = CuicServer.getAPI("192.0.2.207", "1A8DE698E2BF4C0B989476A369F0FC64",
 "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("userid " + instance.getUserId());
  System.out.println("vm id " + instance.getVmId());
  System.out.println("password " + 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 response = instancevix.execute();
  System.out.println("Respose status Code "+response.getResponseErrorCode());
 }
}
```

Result

The response code is returned for the executed VIX script.

Implementation

Use the userAPIExecuteVIXScript API and pass the VM ID, credentials, and VIX script to execute the VIX script on the VM.

Managing VMware System Policy

Creating a VMware System Policy

Objective

Create a VMware system policy to define the system specific information such as the template to use, time zone, OS specific information, and so on for a virtual machine (VM).

Context

To define system specific information for a VM.

Prerequisites

None

REST URL

This section provides the REST URL for creating VMware system policy in different scenarios:

Example 1: Create a VMware system policy with the Linux image

Request

```
/app/api/rest?formatType=json&opName=genericvm:
userAPICreateVMwareSystemPolicy&opData={param0:{"policyName":"test",
"policyDescription":"sample","vmImageType":"Linux Only","vmNameTemplate":"sample",
"vmnamevalidationPolicy":"sample","hostNameTemplate":"sample",
"hostNameValidationPolicy":"sample","linuxVM":{"dnsDomain":"sample",
"dnsSuffix":"sample","dnsServer":"sample","linuxTimeZone":"US/Arizona",
"maxBootWaitTime":10},"windowsVM":"productId":"sample","licenseOwnerName":"sample",
"organizationName":"sample","licenseMode":"sample","noOfLicenseUsers":1000,
"primaryWINs":"sample","secondaryWINS":"sample","maxBootTime":1000,
"isSIDUnique":true,"isAutoLogon":true,"autoLogonCount":1000,
"administratorPassword":"sample","windowsTimezone":"sample","joinTypeDomain":"sample",
"workGroupName":"sample","domain":"sample","domainAdmin":"sample",
"domainPassword":"sample","defineAnnotation":true}}}
```

Response

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"genericvm:userAPICreateVMwareSystemPolicy" }
```

Example 2: Create a VMware system policy with the Windows and Linux image

Request

```
/app/api/rest?formatType=json&opName=genericvm:
userAPICreateVMwareSystemPolicy&opData={param0:{"policyName":"test1",
"policyDescription":"sample","vmImageType":"Windows and Linux",
"vmNameTemplate":"sample","vmnamevalidationPolicy":"sample",
"hostNameTemplate":"sample","hostNameValidationPolicy":"sample",
"linuxVM":{"dnsDomain":"sample","dnsSuffix":"sample","dnsServer":"sample",
"linuxTimeZone":"US/Arizona","maxBootWaitTime":10},"windowsVM":
{"productId":"sample","licenseOwnerName":"sample","organizationName":"sample",
"licenseMode":"Per-Seat","noOfLicenseUsers":1000,"primaryWINs":"sample",
"secondaryWINS":"sample","maxBootTime":10,"isSIDUnique":true,
"isAutoLogon":true,"autoLogonCount":1000,"administratorPassword":"sample",
"windowsTimezone":"Alaskan Time","joinTypeDomain":"Domain",
"workGroupName":"sample","domain":"sample","domainAdmin":"sample",
"domainPassword":"sample","defineAnnotation":true}}}
```

Response

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"genericvm:userAPICreateVMwareSystemPolicy" }
```

Components

None

Code

```
public class CreateSystemPolicyTest
 {
 public static void main(String[] args)throws Exception{
 CuicServer api = CuicServer.getAPI("172.29.110.194", "6BF80FA2C71E4844AFEB3877CFD60621",
 "http", 80);
 UserAPIVMware instance = new UserAPIVMware(api);
 ServiceDeliveryPolicyRequestParam service = new ServiceDeliveryPolicyRequestParam();
 LinuxVMParams linux=new LinuxVMParams();
 linux.setDnsDomain("testdomain");
 linux.setLinuxTimeZone("US/Pacific");
 linux.setLinuxVMmaxBootWaitTime(2);
 WindowsVMParams window=new WindowsVMParams();
 window.setOrganizationName("testOrganization");
 service.setPolicyName("test2");
 service.setPolicyDescription("test");
 service.setVmImageType("Linux Only");
 service.setHostNameTemplate("sample1");
 service.setLinuxVM(linux);
 service.setWindowsVM(window);
 boolean policy= instance.userAPICreateVMwareSystemPolicy(service);
 System.out.println(policy);
 }
```

Results

If the VMware system policy is created successfully, the result is true.

Implementation

Use the userAPICreateVMwareSystemPolicy API and pass the system specific information in the name and value format to create a VMware system policy.

See Also

Retrieving the VMware System Policy Details

Modifying a VMware System Policy

Deleting a VMware System Policy

Retrieving the VMware System Policy Details

Objective

Retrieve the VMware system policy details.

Context

The VMware system policy details are retrieved by the system administrator or the end user.

Prerequisites

None

REST URL

This section provides the REST URL for retrieving VMware system policy in different scenarios:

Example 1: Retrieve details of a specific VMware system policy

Request

```
/app/api/rest?formatType=json&opName=genericvm:
userAPIGetVMwareSystemPolicy&opData=param0:{"policyName":"test",
"policyDescription":"sample","vmImageType":"sample","vmNameTemplate":
"sample","vmnamevalidationPolicy":"sample","hostNameTemplate":"sample",
"hostNameValidationPolicy":"sample","linuxVM":{"dnsDomain":"sample",
"dnsSuffix":"sample","dnsServer":"sample","linuxTimeZone":"sample",
"maxBootWaitTime":0},"windowsVM":{"productId":"sample","licenseOwnerName":
"sample","organizationName":"sample","licenseMode":"sample",
"noOfLicenseUsers":1000,"primaryWINs":"sample","secondaryWINS":"sample",
"maxBootTime":1000,"isSIDUnique":true,"isAutoLogon":true,"autoLogonCount":1000,
"administratorPassword":"sample","windowsTimezone":"sample","
joinTypeDomain":"sample","workGroupName":"sample","domain":"sample",
"domainAdmin":"sample","domainPassword":"sample","defineAnnotation":true}}}
```

Response

```
{ "serviceResult":{"policyId":8,"policyName":"test",
"policyDescription":"sample","vmNameTemplate":"sample",
"vmNameValidationPolicy":"sample","isAllowEndUserSuffix":false,"powerOn":true,
"hostNameTemplate":"sample","hostNameValidationPolicy":"sample",
"linuxTimezone":"US/Arizona","linuxVMMaxBootTime":4,"dnsDomain":"sample",
"dnsSuffix":null,"dnsSuffixList":null,"dnsServer":null,"dnsServerList":null,
"resourcePool":null,"imageType":1,"windowsLabel":null,"productId":null,
"fullName":"CompanyFullName","orgName":"CompanyName","licenseMode":"Per-Seat",
"licenseUsers":5,"winList":null,"secondaryWINS":null,"windowsVMMaxBootTime":10,
"isSIDUnique":false,"isAutoLogon":true,"autoLogonCount":5,"password":null,
"windowsTimezone":4,"joinTypeDomain":true,"workGroupName":null,"domain":null,
"domainAdmin":null,"domainPassword":null,"defineAnnotation":false,"notes":"",
"customAttributes":{"list":[],"moTypeName":
"com.cloupia.service.cIM.inframgr.profiles.PrivateCloudSystemPolicyAnnotation",
"validatorName":null},"customAttributesList":[],"vmAnnotationsClob":null,
"isUseLicenseFromTemplate":false,"skipCustomization":false}, "serviceError":null,
 "serviceName":"InfraMgr", "opName":"genericvm:userAPIGetVMwareSystemPolicy" }
```

Example 2: Retrieve details of all the VMware system policies

Request

```
/app/api/rest?formatType=json&opName=
genericvm:userAPIGetAllVmwareSystemPolicies&opData={}
```

Response

```
{ "serviceResult":[{"policyId":1,"policyName":"Default System Policy",
"policyDescription":"","vmNameTemplate":"vm-${GROUP_NAME}-SR${SR_ID}",
"vmNameValidationPolicy":"","isAllowEndUserSuffix":false,"powerOn":true,
"hostNameTemplate":"${VMNAME}","hostNameValidationPolicy":"",
"linuxTimezone":"US/Pacific","linuxVMMaxBootTime":10,"dnsDomain":"sdk","dnsSuffix":null,
"dnsSuffixList":null,"dnsServer":null,"dnsServerList":null,"resourcePool":"",
"imageType":1,"windowsLabel":null,"productId":"","fullName":"CompanyFullName",
"orgName":"CompanyName","licenseMode":"Per-Seat","licenseUsers":5,"winList":"",
"secondaryWINS":"","windowsVMMaxBootTime":10,"isSIDUnique":false,"isAutoLogon":true,
"autoLogonCount":5,"password":"","windowsTimezone":4,"joinTypeDomain":true,
"workGroupName":"","domain":"","domainAdmin":"","domainPassword":"",
"defineAnnotation":false,"notes":null,"customAttributes":{"list":[],
"moTypeName":"com.cloupia.service.cIM.inframgr.profiles.PrivateCloudSystemPolicyAnnotation",
"validatorName":null},"customAttributesList":[],"vmAnnotationsClob":null,
"isUseLicenseFromTemplate":false,"skipCustomization":false},{"policyId":2,
"policyName":"sample","policyDescription":"sample","vmNameTemplate":"sample",
"vmNameValidationPolicy":"sample","isAllowEndUserSuffix":false,"powerOn":true,
"hostNameTemplate":"host","hostNameValidationPolicy":"sample","linuxTimezone":"US/Arizona",
"linuxVMMaxBootTime":4,"dnsDomain":"sample","dnsSuffix":null,"dnsSuffixList":null,
"dnsServer":null,"dnsServerList":null,"resourcePool":null,"imageType":1,"windowsLabel":null,
"productId":null,"fullName":"CompanyFullName","orgName":"CompanyName","licenseMode":
"Per-Seat","licenseUsers":5,"winList":null,"secondaryWINS":null,"windowsVMMaxBootTime":10,
"isSIDUnique":false,"isAutoLogon":true,"autoLogonCount":5,"password":null,"windowsTimezone":4,
"joinTypeDomain":true,"workGroupName":null,"domain":null,"domainAdmin":null,"domainPassword":null,
"defineAnnotation":false,"notes":"","customAttributes":{"list":[],
"moTypeName":"com.cloupia.service.cIM.inframgr.profiles.PrivateCloudSystemPolicyAnnotation",
"validatorName":null},"customAttributesList":[],"vmAnnotationsClob":null,
"isUseLicenseFromTemplate":false,"skipCustomization":false},{"policyId":3,"policyName":"sample1",
"policyDescription":"sample","vmNameTemplate":"sample","vmNameValidationPolicy":"sample",
"isAllowEndUserSuffix":false,"powerOn":true,"hostNameTemplate":"sample",
"hostNameValidationPolicy":"sample","linuxTimezone":"US/Pacific","linuxVMMaxBootTime":10,
"dnsDomain":null,"dnsSuffix":null,"dnsSuffixList":null,"dnsServer":null,"dnsServerList":null,
"resourcePool":null,"imageType":0,"windowsLabel":null,"productId":"sample","fullName":"sample",
"orgName":"","licenseMode":"Per-Server","licenseUsers":1000,"winList":"sample",
"secondaryWINS":"sample","windowsVMMaxBootTime":10,"isSIDUnique":true,"isAutoLogon":true,
"autoLogonCount":1000,"password":"c2FtcGxl","windowsTimezone":2,"joinTypeDomain":true,
"workGroupName":null,"domain":"admin","domainAdmin":"admin","domainPassword":"YWRtaW4=",
"defineAnnotation":false,"notes":"","customAttributes":{"list":[],
"moTypeName":"com.cloupia.service.cIM.inframgr.profiles.PrivateCloudSystemPolicyAnnotation",
"validatorName":null},"customAttributesList":[],"vmAnnotationsClob":null,
"isUseLicenseFromTemplate":false,"skipCustomization":false},{"policyId":6,"policyName":"sample5",
"policyDescription":"sample","vmNameTemplate":"sample","vmNameValidationPolicy":"sample",
"isAllowEndUserSuffix":false,"powerOn":true,"hostNameTemplate":"sample","hostNameValidationPolicy":
"sample","linuxTimezone":"US/Arizona","linuxVMMaxBootTime":6,"dnsDomain":"sample",
"dnsSuffix":null,"dnsSuffixList":null,"dnsServer":null,"dnsServerList":null,"resourcePool":null,
"imageType":1,"windowsLabel":null,"productId":null,"fullName":"CompanyFullName",
"orgName":"CompanyName","licenseMode":"Per-Seat","licenseUsers":5,"winList":null,
"secondaryWINS":null,"windowsVMMaxBootTime":10,"isSIDUnique":false,"isAutoLogon":true,
"autoLogonCount":5,"password":null,"windowsTimezone":4,"joinTypeDomain":true,"workGroupName":null,
"domain":null,"domainAdmin":null,"domainPassword":null,"defineAnnotation":false,"notes":"",
"customAttributes":{"list":[],
"moTypeName":"com.cloupia.service.cIM.inframgr.profiles.PrivateCloudSystemPolicyAnnotation",
"validatorName":null},"customAttributesList":[],"vmAnnotationsClob":null,
"isUseLicenseFromTemplate":false,"skipCustomization":false},{"policyId":7,"policyName":"sample6",
"policyDescription":"sample","vmNameTemplate":"sample","vmNameValidationPolicy":"sample",
"isAllowEndUserSuffix":false,"powerOn":true,"hostNameTemplate":"sample",
"hostNameValidationPolicy":"sample","linuxTimezone":"US/Pacific","linuxVMMaxBootTime":10,
"dnsDomain":null,"dnsSuffix":null,"dnsSuffixList":null,"dnsServer":null,"dnsServerList":null,
"resourcePool":null,"imageType":0,"windowsLabel":null,"productId":"sample","fullName":"sample",
"orgName":"sample","licenseMode":"Per-Seat","licenseUsers":1000,"winList":"sample",
"secondaryWINS":"sample","windowsVMMaxBootTime":10,"isSIDUnique":true,"isAutoLogon":true,
"autoLogonCount":1000,"password":"c2FtcGxl","windowsTimezone":1,"joinTypeDomain":true,
"workGroupName":null,"domain":"sam","domainAdmin":"sample","domainPassword":"c2FtcGxl",
"defineAnnotation":false,"notes":"","customAttributes":{"list":[],
"moTypeName":"com.cloupia.service.cIM.inframgr.profiles.PrivateCloudSystemPolicyAnnotation",
"validatorName":null},"customAttributesList":[],"vmAnnotationsClob":null,
"isUseLicenseFromTemplate":false,"skipCustomization":false},{"policyId":8,
"policyName":"test","policyDescription":"sample","vmNameTemplate":"sample",
"vmNameValidationPolicy":"sample","isAllowEndUserSuffix":false,"powerOn":true,
"hostNameTemplate":"sample","hostNameValidationPolicy":"sample",
"linuxTimezone":"US/Arizona","linuxVMMaxBootTime":4,"dnsDomain":"sample","dnsSuffix":null,
"dnsSuffixList":null,"dnsServer":null,"dnsServerList":null,"resourcePool":null,"imageType":1,
"windowsLabel":null,"productId":null,"fullName":"CompanyFullName","orgName":"CompanyName",
"licenseMode":"Per-Seat","licenseUsers":5,"winList":null,"secondaryWINS":null,
"windowsVMMaxBootTime":10,"isSIDUnique":false,"isAutoLogon":true,"autoLogonCount":5,
"password":null,"windowsTimezone":4,"joinTypeDomain":true,"workGroupName":null,"domain":null,
"domainAdmin":null,"domainPassword":null,"defineAnnotation":false,"notes":"",
"customAttributes":{"list":[],
"moTypeName":"com.cloupia.service.cIM.inframgr.profiles.PrivateCloudSystemPolicyAnnotation",
"validatorName":null},"customAttributesList":[],"vmAnnotationsClob":null,
"isUseLicenseFromTemplate":false,"skipCustomization":false},{"policyId":9,"policyName":"test1",
"policyDescription":"sample","vmNameTemplate":"sample","vmNameValidationPolicy":"sample",
"isAllowEndUserSuffix":false,"powerOn":true,"hostNameTemplate":"sample",
"hostNameValidationPolicy":"sample","linuxTimezone":"US/Pacific","linuxVMMaxBootTime":10,
"dnsDomain":null,"dnsSuffix":null,"dnsSuffixList":null,"dnsServer":null,"dnsServerList":null,
"resourcePool":null,"imageType":0,"windowsLabel":null,"productId":"sample","fullName":"sample",
"orgName":"sample","licenseMode":"Per-Seat","licenseUsers":1000,"winList":"sample",
"secondaryWINS":"sample","windowsVMMaxBootTime":10,"isSIDUnique":true,"isAutoLogon":true,
"autoLogonCount":1000,"password":"c2FtcGxl","windowsTimezone":3,"joinTypeDomain":true,
"workGroupName":null,"domain":"sample","domainAdmin":"sample","domainPassword":"c2FtcGxl",
"defineAnnotation":false,"notes":"","customAttributes":{"list":[],
"moTypeName":"com.cloupia.service.cIM.inframgr.profiles.PrivateCloudSystemPolicyAnnotation",
"validatorName":null},"customAttributesList":[],"vmAnnotationsClob":null,
"isUseLicenseFromTemplate":false,"skipCustomization":false}], "serviceError":null,
 "serviceName":"InfraMgr", "opName":"genericvm:userAPIGetAllVmwareSystemPolicies" }
```

Components

None

Code

```
public class GetSystemPolicyTest
{
 public static void main(String[] args)throws Exception {
 CuicServer api = CuicServer.getAPI("172.29.110.194", "6BF80FA2C71E4844AFEB3877CFD60621",
 "http", 80);
 UserAPIVMware instance = new UserAPIVMware(api);
 ServiceDeliveryPolicyRequestParam service = new ServiceDeliveryPolicyRequestParam();
 service.setPolicyName("test2");
 PrivateCloudSystemProfile policy= instance.userAPIGetVMwareSystemPolicy(service);
 System.out.println("policyName:" + policy.getPolicyName());
 System.out.println("policyDescription:" +policy.getPolicyDescription());
 System.out.println("ImageType:" + policy.getImageType());
 System.out.println("maxBootTime :" + policy.getLinuxVMMaxBootTime());
}
```

Results

If the valid system policy name is given, the VMware system policy details are retrieved successfully.

The following VMware system policy details are retrieved when calling the userAPIGetVMwareSystemPolicy API:

- policyName:test
- policyDescription: sample
- imageType:Linux Only
- maxBootTime: 10

Implementation

Call the userAPIGetVMwareSystemPolicy API by passing the VMware system policy name to retrieve the VMware system policy details. Call the userAPIGetAllVmwareSystemPolicies API to view details of all the VMware system policies.

See Also

Creating a VMware System Policy

Modifying a VMware System Policy

Deleting a VMware System Policy

Modifying a VMware System Policy

Objective

Update the VMware system policy.

Context

To update the system specific information defined for a VM.

Prerequisites

None

REST URL

Request

```
/app/api/rest?formatType=json&opName=genericvm:
userAPIUpdateVMwareSystemPolicy&opData={param0:{"policyName":"test",
"policyDescription":"sample","vmImageType":"Linux Only",
"vmNameTemplate":"sample","vmnamevalidationPolicy":"sample",
"hostNameTemplate":"sample","hostNameValidationPolicy":"sample","linuxVM":
{"dnsDomain":"sample","dnsSuffix":"sample","dnsServer":"sample",
"linuxTimeZone":"US/Arizona","maxBootWaitTime":4},"windowsVM":
{"productId":"sample","licenseOwnerName":"sample","organizationName":
"sample","licenseMode":"sample","noOfLicenseUsers":1000,"primaryWINs":
"sample","secondaryWINS":"sample","maxBootTime":1000,"isSIDUnique":true,
"isAutoLogon":true,"autoLogonCount":1000,"administratorPassword":"sample",
"windowsTimezone":"sample","joinTypeDomain":"sample","workGroupName":"sample",
"domain":"sample","domainAdmin":"sample","domainPassword":"sample",
"defineAnnotation":true}}}
```

Response

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"genericvm:userAPIUpdateVMwareSystemPolicy" }
```

Components

None

Code

```
public static void main(String[] args)throws Exception
{
  CuicServer api = CuicServer.getAPI("172.29.110.194", "6BF80FA2C71E4844AFEB3877CFD60621",
 "http", 80);
  UserAPIVMware instance = new UserAPIVMware(api);
  ServiceDeliveryPolicyRequestParam service = new ServiceDeliveryPolicyRequestParam();

  service.setPolicyName("test2");
  service.setVmImageType("Linux Only");
  service.setHostNameTemplate("sample2");
  LinuxVMParams linux=new LinuxVMParams();
  linux.setDnsDomain("testdomain");
  linux.setLinuxTimeZone("US/Pacific");
  linux.setLinuxVMmaxBootWaitTime(2);
  WindowsVMParams window=new WindowsVMParams();
  window.setOrganizationName("testOrganization");
  service.setLinuxVM(linux);
  service.setWindowsVM(window);
  boolean policy= instance.userAPIUpdateVMwareSystemPolicy(service);
  System.out.println(policy);
  }
```

Results

If the VMware system policy is updated successfully, the result is true.

Implementation

Use the userAPIUpdateVMwareSystemPolicy API and pass the system specific information that need to be updated in the name and value format to update the VMware system policy.

See Also

Creating a VMware System Policy

Retrieving the VMware System Policy Details

Deleting a VMware System Policy

Deleting a VMware System Policy

Objective

Delete a VMware system policy by passing the policy name.

Context

The VMware system policy can be deleted by a system administrator.

Prerequisites

The VMware system policy to be deleted must exist.

REST URL

Request

```
/app/api/rest?formatType=json&opName=genericvm:
userAPIDeleteVMwareSystemPolicy&opData={param0:{"policyName":"test",
"policyDescription":"sample","vmImageType":"sample","vmNameTemplate":
"sample","vmnamevalidationPolicy":"sample","hostNameTemplate":"sample",
"hostNameValidationPolicy":"sample","linuxVM":{"dnsDomain":"sample",
"dnsSuffix":"sample","dnsServer":"sample","linuxTimeZone":"sample",
"maxBootWaitTime":0},"windowsVM":{"productId":"sample","licenseOwnerName":
"sample","organizationName":"sample","licenseMode":"sample",
"noOfLicenseUsers":1000,"primaryWINs":"sample","secondaryWINS":"sample",
"maxBootTime":1000,"isSIDUnique":true,"isAutoLogon":true,"autoLogonCount":1000,
"administratorPassword":"sample","windowsTimezone":"sample","joinTypeDomain":
"sample","workGroupName":"sample","domain":"sample","domainAdmin":"sample",
"domainPassword":"sample","defineAnnotation":true}}}
```

Response

```
{ "serviceResult":true, "serviceError":null, "serviceName":"InfraMgr",
 "opName":"genericvm:userAPIDeleteVMwareSystemPolicy" }
```

Components

None

Code

```
public class DeleteSystemPolicyTest
{
 public static void main(String[] args) throws Exception]
 {
 CuicServer api = CuicServer.getAPI("172.29.110.194", "6BF80FA2C71E4844AFEB3877CFD60621",
 "http", 80);
 UserAPIVMware instance = new UserAPIVMware(api);
 ServiceDeliveryPolicyRequestParam service = new ServiceDeliveryPolicyRequestParam();
 service.setPolicyName("test23423");
 boolean policy= instance.userAPIDeleteVMwareSystemPolicy(service);
 system.out.println(policy);
}
```

Results

If the VMware system policy is deleted successfully, the result is true.

Implementation

Call the userAPIDeleteVMwareSystemPolicy API by passing the policy name to delete an existing VMware system policy.

See Also

Creating a VMware System Policy

Retrieving the VMware System Policy Details

Modifying a VMware System Policy

Deleting a VMware Snapshot

Objective

Delete a VMware snapshot.

Context

Provide more disk space for newer snapshots. The VMware snapshot can be deleted only by the administrator.

Prerequisites

VMware snapshot must be available.

REST URL

Not Applicable

Components

Snapshot name is required.

Code

```
public class DeleteVMSnapshotExample
{
 public static void main(String[] args)
{
  CuicServer server = CuicServer.getAPI("192.0.2.207", "1A8DE698E2BF4C0B989476A369F0FC64",
 "https", 443);
  DeleteVMSnapshot instance = new DeleteVMSnapshot(server);
  instance.setVmId(168);
  instance.setSnapshotName("snapshot-2015-01-10");
  instance.setDeleteChild(false);
  DeleteVMSnapshotResponse response = instance.execute();
  System.out.println(" Deleted response "+response.getStatus());
  }
}
```

Results

If the VM snapshot is deleted successfully, the result is true.

Implementation

No implementation required.

Managing Workflow Orchestration

Submitting a Service Request

Objective

Submit a service request to execute a workflow to execute a set of operations on the resources such as choosing the VDC on which a VM is provisioned.

Context

Execute a workflow to perform a set of tasks.

Prerequisites

Workflow must be available in Cisco UCS Director. User must be authorized to execute the workflow.

REST URL

```
/app/api/rest?formatType=json&opName=userAPISubmitServiceRequest&opData=
{param0:"testCatalog",param1:"vdc1",param2:1,param3:-1,param4:1,param5:"provisioning
vm"}
```

Components

Workflow name is required.

Code

```
public class UserAPISubmitServiceRequestExample
{
 public static void main(String[] args) throws Exception{
 CuicServer server = CuicServer.getAPI("192.0.2.207", "1A8DE698E2BF4C0B989476A369F0FC64",
 "https", 443);
 UserAPIGlobal instance = new UserAPIGlobal(server);
 int srId = instance.userAPISubmitServiceRequest("testCatalog", "vdc", 1, -1, 1,
"UCSD-5.4.0.0");
 System.out.println("srId "+srId);
     }
}
```

Results

Service request ID is returned.

Implementation

Call the userAPISubmitServiceRequest API and pass the workflow name to execute a workflow.

See Also

Submitting a VApp Request

Retrieving Output of a Service Request

Rollback a Workflow

Retrieving Log Entries of a Service Request

Submitting a VApp Request

Objective

Submit a service request with the advanced catalog type and arguments.

Context

Execute a workflow for advanced catalog type.

Prerequisites

Advanced catalog must be available in Cisco UCS Director.

REST URL

```
/app/api/rest?formatType=json&opName=userAPISubmitVAppServiceRequest&opData={param0:
"PjaCat",param1:{"list":[{"name":"Tenant Name","value":"Pja_27"},{"name":"Tenant
Description",
"value":"none"},{"name":"MSP Admin","value":"asa"},{"name":"MSP Admin
Password","value":"asa"},
{"name":"MSP Admin Email","value":"asa"},{"name":"Datastore Size(GB)","value":"10"},
{"name":"Memory Reservation(MB)","value":"100"},{"name":"No of CPU","value":"5"},{"name":
"No of VDCs","value":"5"},{"name":"L2 Or L3 External Network
Configuration","value":"None"},
{"name":"L2 VLAN ID","value":""},{"name":"L2 IP Subnet (x.x.x.x/n)","value":""},{"name":
"Tenant IP Subnet (x.x.x.x/n)","value":"10.12.18.0/16"},{"name":"Replication
Required","value":
"No"}]}}
```

Components

Advanced catalog name

Code

```
public class UserAPISubmitVAppServiceRequestExample
{
 public static void main(String[] args) throws Exception
{
  CuicServer server = CuicServer.getAPI("192.0.2.207", "1A8DE698E2BF4C0B989476A369F0FC64",
 "https", 443);
  UserAPIGlobal instance = new UserAPIGlobal(server);
  APINameValueList list = new APINameValueList();
  APINameValue nv = new APINameValue();
  nv.setName("Tenant Name");
  nv.setValue("Tenant1");
  nv.setName("Tenant Description");
  nv.setValue("");
  nv.setName("Tenant Group");
  nv.setValue("MSPGroup");
  list.addNameValue(nv);
  int srId = instance.userAPISubmitVAppServiceRequest("ExecuteAdvanceCat", list);
  System.out.println("srId "+srId);
  }
}
```

Results

The service request ID is returned.

Implementation

Call the userAPISubmitVAppServiceRequest API and pass the advanced catalog name to execute a workflow for advanced catalog type.

See Also

Submitting a Service Request

Retrieving Output of a Service Request

Rollback a Workflow

Retrieving Log Entries of a Service Request

Retrieving Output of a Service Request

Objective

Retrieve the output details of a service request by passing the service request ID.

Context

To know the output of a service request.

Prerequisites

A successfully executed service request must be available in Cisco UCS Director. Record the service request ID.

REST URL

Request

```
/app/api/rest?formatType=json&opName=servicerequest:
userAPIGetServiceRequestOutputDetails&opData={param0:1}
```

Response

```
{ "serviceResult":{"workflowOutputDetails":[]}, "serviceError":null,
"serviceName":"InfraMgr",
 "opName":"servicerequest:userAPIGetServiceRequestOutputDetails" }
```

Components

param0—The ID of the service request for which you want to view the output.

Code

```
import java.util.List;

import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.UserAPIServiceRequest;
import com.cisco.cuic.api.models.UserAPIVMware;
import com.cisco.cuic.api.models.servicerequest.APIWorkflowOutputDetails;
import com.cisco.cuic.api.models.servicerequest.APIWorkflowOutputFieldDetails;
import com.cisco.cuic.api.models.vmware.ServiceDeliveryPolicyRequestParam;

public class ServiceRequestOutputDetailsTest {

 public static void main(String[] args)throws Exception {
  CuicServer api = CuicServer.getAPI("172.22.234.243", "CF87FA987C8F4BBF814F2BB68CA6A823",
 "http", 80);

  UserAPIServiceRequest instance = new UserAPIServiceRequest(api);
  APIWorkflowOutputDetails outputDetails =
instance.userAPIGetServiceRequestOutputDetails(1);
  List<APIWorkflowOutputFieldDetails> outputList
=outputDetails.getWorkflowOutputDetails();

  String outputFieldName=outputList.get(0).getOutputFieldName();
  String outputFieldType=outputList.get(0).getOutputFieldType();
  String ouputFieldDescription=outputList.get(0).getOutputFieldDescription();
  String ouputFieldValue=outputList.get(0).getOutputFieldValue();

  System.out.println(outputFieldName);
  System.out.println(outputFieldType);
  System.out.println(outputFieldType);
  System.out.println(ouputFieldValue);
 }
 }
```

Results

The output details of the service request is displayed.

Implementation

Call the userAPIGetServiceRequestOutputDetails API and pass the service request ID to view the output of the service request.

See Also

Submitting a Service Request

Submitting a VApp Request

Rollback a Workflow

Retrieving Log Entries of a Service Request

Rollback a Workflow

Objective

Roll back a workflow to undo a specific operation. A service request ID is generated on successful rollback of the workflow. A system administrator or an end user can roll back a workflow. The end user can roll back a service request only when the user role has the Write - Group Service Request user permission.

If a user rolls back a service request initiated by other user, a rollback workflow approval is triggered to get approval from the service request initiated user. The rollback workflow is executed after getting approval from the service request initiated user.

Note If a workflow with one or more compound workflow tasks is rolled back, only one rollback service request ID is generated. The child service requests are not generated for the compound workflow tasks.

Context

Undo the workflow operation.

Prerequisites

A successfully executed service request must be available in Cisco UCS Director. Record the service request ID.

REST URL

```
/app/api/rest?formatType=json&opName=userAPIRollbackWorkflow&opData={param0:40}
```

Components

int srId—The service request ID of the workflow that you want to roll back.

Code

```
public class UserAPIRollbackWorkflowExample
{
 public static void main(String[] args) throws Exception
 {
  CuicServer server = CuicServer.getAPI("192.0.2.207", "1A8DE698E2BF4C0B989476A369F0FC64",
 "https", 443);
  UserAPIGlobal instance = new UserAPIGlobal(server);
  int srId = instance.userAPIRollbackWorkflow(123);
  System.out.println("srId " + srId);
  }
}
```

Results

The service request ID is returned.

Implementation

Call the userAPIRollbackWorkflow API and pass the service request ID to roll back a service request.

See Also

Submitting a Service Request

Submitting a VApp Request

Retrieving Output of a Service Request

Retrieving Log Entries of a Service Request

Retrieving Log Entries of a Service Request

Objective

Retrieve the log details of a service request by passing the service request ID.

Context

To view the log entries of a service request.

Prerequisites

A successfully executed service request must be available in Cisco UCS Director. Record the service request ID.

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPIGetServiceRequestLogEntries&opData=
{param0:59,param1:0}
```

Response

```
{"ServiceResult":[{"srId""59,"timestamp":1489738775609,"severity":0,
"message":"Executing workflow item number
1","indent":0},{"srId":59,"timestamp":1489738775619,
"severity":0,"message":"Completed workflow item number 1, with status
completed","indent":0}
{"srId":59,"timestamp":1489738778617, "severity":0,"message":"Executing workflow item
number 2","indent":0}
{"srId":59,"timestamp":1489738778668,"severity":0,"message":"Executing custom action
WFCloupiaScript

(custom_GlobalIdenAPIsTest))","indent":0}{"srId":59,"timestamp":1489738779075,"severity":0,
"message":"Input/Output values for Task = 2
(WFCloupiaScript(custom_GlobalIdenAPIsTest))","indent":0}
{"srId":59,"timestamp":1489738779077,"severity":0,"message":"Completed workflow item
number 2, with status Completed",
"indent":0}{"srId":59,"timestamp":1489738781625,"severity":0,"message":"Executing
wofkflow item number 3","imdent":0}
{"srId":59,"timestamp":1489738781630,"severity":0,"message":"Executing wofkflow item
number 3, with status Completed",
"indent":0}], "serviceError":null,
"serviceName":"InfraMgr","opName":"userAPIGetServiceReqeustLogEntries"}
```

Components

- param0—The ID of the service request for which you want to view the log details.
- param1—The serverity ID of the service request. The severity ID can be:

- 0—SEVERITY_DEBUG
- 1—SEVERITY_INFO
- 2—SEVERITY_WARNING
- 3—SEVERITY_ERROR

Code

```
public class GetServiceRequestLogEntries {
public static void main(String[] args) throws Exception {
CuicServer api = CuicServer.getAPI("10.23.210.42", "3E17CFFBA7A64C71B8958F40DE2EC9B3",
 "http", 80);
UserAPIGlobal instance = new UserAPIGlobal(api);
//Pass the service request ID and severity (0,1,2,3)
ServiceRequestLogEntry[] entries = instance.userAPIGetServiceRequestLogEntries(1, 0);
for (int i = 0; i < entries.length; i++) {
    ServiceRequestLogEntry entry = entries[i];
    System.out.println("log timestamp: " + entry.getTimestamp());
    System.out.println("log message: " + entry.getMessage());
}
}
}
```

Results

The log details of the passed service request is displayed.

Implementation

Call the userAPIGetServiceRequestLogEntries API and pass the ID and severity ID of the service request to retrieve the log entries of the service request.

See Also

Submitting a Service Request

Submitting a VApp Request

Retrieving Output of a Service Request

Rollback a Workflow

Managing Import and Export of Cisco UCS Director Artifacts

You can export and import workflows, custom tasks, script modules, and activities in Cisco UCS Director. This is useful if, for example, you want to:

- Move or copy workflows or other entities to different Cisco UCS Director instances.
- Back up or store entities.
- Use templates to standardize workflows.

Cisco UCS Director artifacts are exported from and imported to Cisco UCS Director in a single package. The exported or imported file has a.wfdx extension and is an XML file containing a serialized representation of the objects.

General Guidelines

- Any special characters (say, for example, “) must be URL encoded. In the examples captured in the document, %22 is used.

  Note The URL is enclosed within single quotes to avoid parsing by the shell.

- The REST API Access Key must be copied from Cisco UCS Director. For more information, refer the Generating an API Access Key section in the *Cisco UCS Director REST API Getting Started Guide*.
- The userAPIDownloadFile API, which is used to download a file, returns Base64 encoded data. You must save the encoded data into a file. See Downloading a File with the Exported Artifacts, on page 82.

Exporting Artifacts

Objective

Export one or more artifacts from the Cisco UCS Director server to the local system using the userAPIUnifiedExport API. After initiating the artifact export operation, check for the export status of artifact using the userAPIUnifiedExportStatus API. For more information, see Viewing the Export Status of Artifacts, on page 81.The exported file is saved in the `/opt/infra/web_cloudmgr/apache-tomcat/webapps/app/cloudmgr/exports` folder.

Once the export is completed, download the exported file to the local system using the userAPIDownloadFile API. For more information, see Downloading a File with the Exported Artifacts, on page 82.

The exported file has a .wfdx extension and is an XML file containing a serialized representation of the objects.

The file contains at least one of the following:

- One or more workflows
- One or more custom tasks
- One or more script modules
- One or more activities

Context

You can export the artifacts of Cisco UCS Director to the local system to move or copy workflows or other entities to different Cisco UCS Director instances. You can export artifacts to take back up or store entities in the local system.

You can take a copy of the .wfdx file from the `exports` folder and use the .wfdx file as required.

Prerequisites

The REST API must be called with an admin user ID.

REST URL

Request

```
/app/api/rest?formatType=json&opName=version:userAPIUnifiedExport&opData=
{param0:{%22workflowIds%22:[%22450%22,%22442%22],%22customTaskIds%22:[],
%22scriptModuleId%22:[],%22activityNames%22:[]},param1:%22TestExport%22}
```

Response

```
{ "serviceResult":"TestExport.wfdx", "serviceError":null, "serviceName":"InfraMgr",
"opName":"version:userAPIUnifiedExport" }
```

The response indicates that Cisco UCS Director has started the process of exporting the requested artifacts into the `TestExport.wfdx` file.

You have to use the file name in the response to check the export status and to download the exported file.

Components

The parameters of the userAPIUnifiedExport API are:

- String workflowIds—Optional. An array of workflow IDs that you want to export. If you dont want to export a workflow along with other artifacts, leave this string empty.
- customTaskIds—Optional. An array of custom tasks that you want to export. If you dont want to export a custom task along with other artifacts, leave this string empty.
- scriptModuleId—Optional. An array of script modules that you want to export. If you dont want to export a script module along with other artifacts, leave this string empty.
- activityNames—Optional. An array of activity names that you want to export. If you dont want to export an activity along with other artifacts, leave this string empty.
- param1—The name of the file into which the contents have to be exported. Specify only the file name, the file path is not required. If the filename has any special characters, encode them with %XX%.

  As per the REST URL, the workflows with IDs [450, 442] are exported into a `TestExport` file.

Code

Not applicable

Results

The artifacts are exported to the local system.

Implementation

No modifications required to implement this code.

See Also

- Viewing the Export Status of Artifacts
- Downloading a File with the Exported Artifacts
- Importing Artifacts
- Viewing the Import Status of Artifacts
- Uploading Files to Cisco UCS Director
- Viewing the Status of Files Uploaded into Cisco UCS Director

Viewing the Export Status of Artifacts

Objective

To check the progress of the artifact export initiated using the userAPIUnifiedExport API.

Context

After raising a request to export artifacts using the userAPIUnifiedExport API, you can check the progress of creation of the wfdx file that is used for exporting the artifacts.

Prerequisites

- The REST API must be called with an admin user ID.
- The userAPIUnifiedExportStatus API must be triggered only after exporting the artifacts using the userAPIUnifiedExport API.

REST URL

Request

```
/app/api/rest?formatType=json&opName=
version:userAPIUnifiedExportStatus&opData={param0:"TestExport.wfdx"}
```

Response

```
{ "serviceResult":"Completed", "serviceError":null, "serviceName":"InfraMgr",
 "opName":"version:userAPIUnifiedExportStatus" }
```

The response indicates that the export operation is completed.

Components

The parameters of the userAPIUnifiedExportStatus API are:

- param0—The name of the exported workflow file with the .wfdx extension.

The following example shows the request and response of REST URL when you pass a wrong file name or file name without the .wfdx extension.

Request

```
/app/api/rest?formatType=json&opName=version:userAPIUnifiedExportStatus&opData=
{param0:"TestExport"}
```

Response

```
{ "serviceResult":"File:TestExport Alone not found, kindly retry the export",
"serviceError":null, "serviceName":"InfraMgr",
"opName":"version:userAPIUnifiedExportStatus" }
```

Code

Not applicable

Results

One of the following option is displayed as the status of the exported workflow file:

- In progress
- Completed
- Failed

Implementation

No modifications required to implement this code.

See Also

- Exporting Artifacts
- Downloading a File with the Exported Artifacts
- Importing Artifacts
- Viewing the Import Status of Artifacts
- Uploading Files to Cisco UCS Director
- Viewing the Status of Files Uploaded into Cisco UCS Director

Downloading a File with the Exported Artifacts

Objective

Download a file with the exported artifacts from the Cisco UCS Director server.

Context

You can download a file with the exported artifacts from Cisco UCS Director using the userAPIDownloadFile API. Then, you can reuse the exported workflow file in another or the latest release of Cisco UCS Director.

Prerequisites

- The REST API must be called with an admin user ID.
- The workflow file that you want to download, must be saved in the `exports` folder.

  If the workflow file is not available in the server path, the file download operation fails and retuns the following response:

  ```
  { "serviceResponse":null, "serviceError":"There is no file with the name 'Wf1' for
   download.",
   "serviceName":"userAPIDownloadFile", "opName":"InfraMgr" }
  ```

REST URL

Request

```
/app/api/rest?formatType=json&opName=
version:userAPIDownloadFile&opData={param0:%22TestExport.wfdx%22}
```

Response

The content of the workflow is displayed in the **Response** field, as a multipart attachment. The data is in the XML format and a snippet of the file is as follows:

```
<?xml version="1.0" ?><OrchExportInfo><Time>Thu Mar 15 09:37:42 PDT
2018</Time><User></User>
<Comments></Comments><UnifiedFeatureAssetInfo><addiInfo></addiInfo><featureAssetEntry
```

To save the file, you have to extract the content between the tags <xml version="1.0?> and …</OrchExportInfo> using Linux tools such as grep and sed as follows:

```
/app/api/rest?formatType=json&opName=
version:userAPIDownloadFile&opData={param0:%22TestExport.wfdx%22}
| grep -e '<?xml version=.*</OrchExportInfo>' > MyDownloadFile.wfdx
```

Components

The parameters of the userAPIDownloadFile API are:

- param0—The name of the workflow file with the .wfdx extension. The file name must not include any path or drive name.

Code

Not applicable

Results

If the file is downloaded successfully, the result is true.

Implementation

No modifications required to implement this code.

See Also

- Exporting Artifacts
- Viewing the Export Status of Artifacts
- Importing Artifacts
- Viewing the Import Status of Artifacts
- Uploading Files to Cisco UCS Director
- Viewing the Status of Files Uploaded into Cisco UCS Director

Importing Artifacts

Objective

Import the contents of exported workflow file into the Cisco UCS Director server through a third party tool, such as, Postman.

Context

If you want to upload and also import the contents of an exported file into Cisco UCS Director, use the userAPIUnifiedImport API in a third party tool, scuh as, Postman.. The userAPIUnifiedImport handles uploading as well as importing in one call.

To execute the userAPIUnifiedImport API through Postman:

1. Log in to Cisco UCS Director.
2. Choose **Orchestration**.
3. Click **REST API Browser**.
4. Enter userAPIUnifiedImport in the search field and double-click the userAPIUnifiedImport API. Pass the required parameters and click **Generate URL**. Copy the URL.

5. Copy the REST API Access Key. For more information, refer the *Generating an API Access Key* section in the *Cisco UCS Director REST API Getting Started Guide*.
6. Log in to Postman.
7. Choose **POST**.
8. Paste the URL copied from Cisco UCS Director (refer Step 4) along with the IP address.

   Sample URL:

   ```
   /app/api/rest?formatType=json&opName=
   version:userAPIUnifiedImport&opData={param0:{"uploadPolicy":"keep
   both","description":"sample"}
   ,param1:{"workflowImportPolicy":"skip","customTaskImportPolicy":"skip",
   "ScriptModuleImportPolicy":"skip","activityImportPolicy":"skip"},param2:"Sample_API"}
   ```
9. Click **Headers**. Enter the Key Name (X-Cloupia-Request-Key) and the Rest API Access Key Value (refer Step 5).
10. Click **Body**. Click the **form-data** radio button. Choose **File** from the **Key** column and click **Choose Files** to select the workflow file (that has to be imported) from the local system.
11. Click **Send** to initiate the file import operation.
12. Check the file import status. For more information, see Viewing the Status of Files Uploaded into Cisco UCS Director, on page 91.

    The imported file is saved in the `/opt/infra/uploads/ApiUploads/` folder.

    The sample response of Postman:

    ```
    { "serviceResult":"TestWorkflows.wfdx", "serviceError":null,
    "serviceName":"InfraMgr",
     "opName":"version:userAPIUnifiedImport" }
    ```

Prerequisites

The REST API must be called with an admin user ID.

REST URL

Request

```
/app/api/rest?formatType=json&opName=
version:userAPIUnifiedImport&opData={param0:{"uploadPolicy":"keep
both","description":"sample"}
,param1:{"workflowImportPolicy":"skip","customTaskImportPolicy":"skip",
"ScriptModuleImportPolicy":"skip","activityImportPolicy":"skip"},param2:"Sample_API"}
```

Response

```
{ "serviceResult":"TestWorkflows.wfdx", "serviceError":null, "serviceName":"InfraMgr",

 "opName":"version:userAPIUnifiedImport" }
```

Components

The parameters of the userAPIUnifiedImport API are:

- uploadPolicy—If the name of the uploaded file does not exist in the `ApiUploads` folder of the server, the current upload process saves the file in the `ApiUploads` folder. If the uploaded file is already available in the `ApiUploads` folder, the file is saved as per the one of the following value of uploadpolicy:

    - keep both—To keep both the files in the `ApiUploads` folder. The latest file is saved with the time stamp in the following format: <uploaded file name_time stamp>.

    Note keep both must be entered with a space between keep and both as two words.

    - Replace—To replace the existing file with the latest file.

    Note The skip value is not applicable for the uploadPolicy.

- Description—Optional. The description of the workflow.
- workflowImportPolicy—If the name of the imported workflow does not exist in Cisco UCS Director and the `ApiUploads` folder of the server, the workflow is saved in Cisco UCS Director and the `ApiUploads` folder. If the imported workflow is already available in Cisco UCS Director and the `ApiUploads` folder, the workflow is saved as per the one of the following value of workflowImportPolicy:
    - skip—To skip the import process without any notification.
    - keep both—To keep both the workflows in Cisco UCS Director and the `ApiUploads` folder. The latest file is saved with the time stamp in the `ApiUploads` folder in the following format: <workflow name_time stamp>. The latest imported workflow is displayed in the Workflow page of Cisco UCS Director and the older version of the workflow is saved at the back end.

    Note keep both must be entered with a space between keep and both as two words.

    - Replace—To replace the existing workflow with the latest workflow.

    Note You must choose at least one of the workflow import policy to import the workflow into Cisco UCS Director and in the `ApiUploads` folder. If you want to import custom task, script module, or activites without workflow, you must pass the empty value as `"workflowImportPolicy":""`.

- customTaskImportPolicy—If the name of the imported custom task does not exist in Cisco UCS Director and the `ApiUploads` folder of the server, the custom task is saved in Cisco UCS Director and the `ApiUploads` folder. If the imported custom task is already available in Cisco UCS Director and the `ApiUploads` folder, the custom task is saved as per the one of the following value of customTaskImportPolicy:
    - skip—To skip the import process without any notification.
    - keep both—To keep both the custom tasks in Cisco UCS Director and the `ApiUploads` folder. The latest file is saved with the time stamp in the `ApiUploads` folder in the following format: <custom task name_time stamp>. The latest imported custom task is displayed in the Custom

Workflow Tasks page of Cisco UCS Director and the older version of the custom task is saved at the back end.

Note keep both must be entered with a space between keep and both as two words.

- Replace—To replace the existing custom task with the latest custom task.

Note You must choose at least one of the custom task import policy to import the custom task into Cisco UCS Director and in the `ApiUploads` folder. If you want to import workflow, script module, or activites without custom task, you must pass the empty value as `"customTaskImportPolicy":""`.

- ScriptModuleImportPolicy—If the name of the imported script module does not exist in Cisco UCS Director and the `ApiUploads` folder of the server, the script module is saved in Cisco UCS Director and the `ApiUploads` folder. If the imported script module is already available in Cisco UCS Director and the `ApiUploads` folder, the script module is saved as per the one of the following value of ScriptModuleImportPolicy:
  - skip—To skip the import process without any notification.
  - keep both—To keep both the script modules in Cisco UCS Director and the `ApiUploads` folder. The latest file is saved with the time stamp in the `ApiUploads` folder in the following format: <script module name_time stamp>. The latest imported script module is displayed in the Script Modules page of Cisco UCS Director and the older version of the script module is saved at the back end.

Note keep both must be entered with a space between keep and both as two words.

  - Replace—To replace the existing script module with the latest script module.

Note You must choose at least one of the script module import policy to import the script module into Cisco UCS Director and in the `ApiUploads` folder. If you want to import workflow, custom task, or activites without script module, you must pass the empty value as `"ScriptModuleImportPolicy":""`.

- activityImportPolicy—If the name of the imported activity does not exist in Cisco UCS Director and the `ApiUploads` folder of the server, the activity is saved in Cisco UCS Director and the `ApiUploads` folder. If the imported activity is already available in Cisco UCS Director and the `ApiUploads` folder, the activity is saved as per the one of the following value of activityImportPolicy:
  - skip—To skip the import process without any notification.
  - keep both—To keep both the activities in Cisco UCS Director and the `ApiUploads` folder. The latest file is saved with the time stamp in the `ApiUploads` folder in the following format:

<activitye name_time stamp>. The latest imported activity is displayed in the Activites page of Cisco UCS Director and the older version of the activity is saved at the back end.

Note keep both must be entered with a space between keep and both as two words.

- Replace—To replace the existing activity with the latest activity.

Note You must choose at least one of the activity import policy to import the activity into Cisco UCS Director and in the `ApiUploads` folder. If you want to import workflow, custom task, or script module without activities, you must pass the empty value as `"activityImportPolicy":""`.

- param2—The name of the folder into which the file contents (workflows, custom tasks, script modules, and activites) has to be imported.

Code

Not applicable

Results

If the artifact is imported successfully, the result is true.

Implementation

No modifications required to implement this code.

See Also

- Exporting Artifacts
- Viewing the Export Status of Artifacts
- Downloading a File with the Exported Artifacts
- Viewing the Import Status of Artifacts
- Uploading Files to Cisco UCS Director
- Viewing the Status of Files Uploaded into Cisco UCS Director

Viewing the Import Status of Artifacts

Objective

View the status of artifacts imported into the Cisco UCS Director server.

Context

After initiating the file import operation using the userAPIUnifiedImport API, you can check the status of files that are imported into the Cisco UCS Director server using the userAPIUnifiedImportStatus API.

Prerequisites

The REST API must be called with an admin user ID.

REST URL

Request

```
/app/api/rest?formatType=json&opName=
version:userAPIUnifiedImportStatus&opData={param0:%22TestWorkflows.wfdx%22}
```

Response

The response indicates a successful import of the file. It also indicates that some elements (custom inputs) are skipped as they already exists in Cisco UCS Director.

```
{ "serviceResult":"number - these custom input(s) skipped since they already
exist import completed", "serviceError":null, "serviceName":"InfraMgr",
"opName":"version:userAPIUnifiedImportStatus" }
```

Components

The parameter of the userAPIUnifiedImportStatus API is:

- param0—The name of the imported workflow file with the .wfdx extension.

The following example shows the request and response of REST URL when you pass a wrong file name or file name without the .wfdx extension.

Request

```
/app/api/rest?formatType=json&opName=
version:userAPIUnifiedImportStatus&opData={param0:%22TestWorkflows%22}
```

Response

```
{ "serviceResult":"TestWorkflows :File Not Found, Please Retry", "serviceError":null,
 "serviceName":"InfraMgr", "opName":"version:userAPIUnifiedImportStatus" }
```

Code

Not applicable

Results

One of the following option is displayed as the status of the imported workflow file:

- In progress
- Completed
- Failed

Implementation

No modifications required to implement this code.

See Also

- Exporting Artifacts
- Viewing the Export Status of Artifacts
- Downloading a File with the Exported Artifacts
- Importing Artifacts

Uploading Files to Cisco UCS Director

Objective

Upload files with multipart atachment (downloaded from Cisco UCS Director using the userAPIDownloadFile API) into the Cisco UCS Director server through a third party tool, such as, Postman.

Context

If you want to upload a file with multipart attachment into Cisco UCS Director from your local system, use the userAPIUploadFile API in a third party tool, such as, Postman. If you want to upload and then import the contents of an exported file into Cisco UCS Director, use the userAPIUnifiedImport API. The userAPIUnifiedImport handles uploading as well as importing in one call. For more information, see Importing Artifacts, on page 83.

Note Do not specify the Content-Type header of the userAPIUploadFile API as `application/json`. Specify the local file that must be uploaded to Cisco UCS Director, in the –F argument. The parameters do not have any significance as the multipart attachment has all the details.

To execute the userAPIUploadFile API through Postman:

1. Log in to Cisco UCS Director.
2. Choose **Orchestration**.
3. Click **REST API Browser**.
4. Enter userAPIUploadFile in the search field and double-click the userAPIUploadFile API. Pass the required parameters and click **Generate URL**. Copy the URL.
5. Copy the REST API Access Key. For more information, refer the *Generating an API Access Key* section in the *Cisco UCS Director REST API Getting Started Guide*.
6. Log in to Postman.
7. Choose **POST**.
8. Paste the URL copied from Cisco UCS Director (refer Step 4) along with the IP address.

   Sample URL:

   ```
   /app/api/rest?formatType=json&opName=servicerequest:userAPIUploadFile&opData={param0:
   {%22uploadPolicy%22:%22replace%22,%22description%22%22:%22TestWorkflows.wfdx%22}}
   ```
9. Click **Headers**. Enter the Key Name (X-Cloupia-Request-Key) and the Rest API Access Key Value (refer Step 5).
10. Click **Body**. Click the **form-data** radio button. Choose **File** from the **Key** column and click **Choose Files** to browse the file (that has to be uploaded) from the local system.
11. Click **Send** to initiate the file upload operation.

12. Check the file import status. For more information, see Viewing the Status of Files Uploaded into Cisco UCS Director, on page 91.

    The imported file is saved in their specified folders and in the `ApiUploads` folder.

    The sample response of Postman:

```
{ "serviceResult":{"fileName":"TestWorkflows.wfdx","description":"","comment":null,
"uploadPolicy":"","fileSize":36079,"fileUploadStatus":"Started","failureReason":"",
"userName":"admin","isFileExist":false}, "serviceError":null,
"serviceName":"InfraMgr",
 "opName":"servicerequest:userAPIUploadFile" }
```

Prerequisites

The REST API must be called with an admin user ID.

REST URL

Request

```
/app/api/rest?formatType=json&opName=servicerequest:userAPIUploadFile&opData={param0:
{%22uploadPolicy%22:%22replace%22,%22description%22%22:%22TestWorkflows.wfdx%22}}
```

Response

The response provides the uploaded file size and the current file upload status. It also indicates if a workflow file with the same name already exists.

```
{ "serviceResult":{"fileName":"TestWorkflows.wfdx","description":"","comment":null,
"uploadPolicy":"","fileSize":36079,"fileUploadStatus":"Started","failureReason":"",
"userName":"admin","isFileExist":false}, "serviceError":null, "serviceName":"InfraMgr",

 "opName":"servicerequest:userAPIUploadFile" }
```

Components

The parameters of the userAPIUploadFile API are:

- uploadPolicy—If the name of the uploaded file does not exist in the `ApiUploads` folder of the server, the current upload process saves the file in the `ApiUploads` folder. If the uploaded file is already available in the `ApiUploads` folder, the file is saved as per the one of the following value of uploadpolicy:
    - skip—To skip the upload process without any notification.
    - keep both—To keep both the files in the `ApiUploads` folder. The latest file is saved with the time stamp in the following format: <uploaded file name_time stamp>.

    Note keep both must be entered with a space between keep and both as two words.

    - Replace—To replace the existing file with the latest file.
- Description—Optional. The description of the file.

Code

Not applicable

Results

If the file is uploaded successfully, the result is true.

Implementation

No modifications required to implement this code.

See Also

- Exporting Artifacts
- Viewing the Export Status of Artifacts
- Downloading a File with the Exported Artifacts
- Importing Artifacts
- Viewing the Import Status of Artifacts
- Viewing the Status of Files Uploaded into Cisco UCS Director

Viewing the Status of Files Uploaded into Cisco UCS Director

Objective

View the status of file uploaded into the Cisco UCS Director server.

Context

After uploading a file using the userAPIUploadFile API, you can check the progress or upload status of the file that are uploaded to the Cisco UCS Director server using the userAPIGetFileUploadStatus API.

Prerequisites

The REST API must be called with an admin user ID.

REST URL

Request

```
/app/api/rest?formatType=json&opName=
servicerequest:userAPIGetFileUploadStatus&opData={param0:%22TestWorkflows.wfdx%22}
```

Response

The following response indicates that the uploading operation of the `TestWorkflows.wfdx` file is completed.

```
{
"serviceResult":{"fileName":"TestWorkflows.wfdx","uploadStatus":"Completed","failureCause":""},

 "serviceError":null, "serviceName":"InfraMgr",
"opName":"servicerequest:userAPIGetFileUploadStatus" }
```

Components

The parameter of the userAPIGetFileUploadStatus API is:

- param0—The name of the uploaded file with the .wfdx extension.

Code

Not applicable

Results

One of the following option is displayed as the status of the uploaded workflow file:

- In progress
- Completed
- Failed

Implementation

No modifications required to implement this code.

See Also

- Exporting Artifacts
- Viewing the Export Status of Artifacts
- Downloading a File with the Exported Artifacts
- Importing Artifacts
- Viewing the Import Status of Artifacts
- Uploading Files to Cisco UCS Director

Retrieving Workflow Fields

Retrieving Input Fields of a Workflow Associated with a Catalog

Objective

Retrieve the input fields of a workflow that is associated with an advanced catalog. You can view the input fields such as input label, name, description, input type, field type, and catalog type of a workflow.

Context

To know the input fields of the workflow associated with the advanced catalog.

Prerequisites

Catalog must be associated with a workflow.

REST URL

Request

```
/app/api/rest?formatType=json&opName=catalog:
userAPIGetCatalogInputDefinition&opData={param0:{"catalogName":"AdvCatalog"}}
```

Response

```
{ "serviceResult":{"details":[{"name":"input_0_input1989","label":"input1",
"description":"","type":"gen_text_input","catalogType":null,"isOptional":false,
"inputFieldType":"text","isMultiSelect":false,"inputFieldValidator":null}]},
 "serviceError":null, "serviceName":"InfraMgr",
"opName":"catalog:userAPIGetCatalogInputDefinition" }
```

Components

catalogName—Name of the advanced catalog.

Code

```
public class GetCatalogInputDef
{
  public static void main(String[] args) throws Exception {
  CuicServer api = CuicServer.getAPI("10.23.210.42", "3E17CFFBA7A64C71B8958F40DE2EC9B3",
 "http", 80);
  UserAPICatalog catalog = new UserAPICatalog(api);
  APICatalogParams params = new APICatalogParams();
  params.setCatalogName("AdvCatalog");
  APIWorkflowInputDetails details = catalog.userAPIGetCatalogInputDefinition(params);
  List<APIWorkflowInputDetail> list = null;
  if(details != null){
   list = details.getDetails();
  }else{
   throw new Exception("No input defination found!");
  }
  if(list != null && list.size() > 0){
  for(int i=0;i<list.size();i++){
   System.out.println("Field Name: "+list.get(i).getName());
   System.out.println("Field Label: "+list.get(i).getLabel());
   System.out.println("Field Description: "+list.get(i).getDescription());
   System.out.println("Input Type: "+list.get(i).getType());
   System.out.println("Field Type: "+list.get(i).getInputFieldType());
   System.out.println("Catalog Type: "+list.get(i).getCatalogType());
  }
  }else{
   System.out.println("No catalog input defination found!");
  }
 }
}
```

Results

The input fields of the workflow associated with the catalog are listed.

Sample Result

- Field Name: input_0_input1989
- Field Label: input1
- Field Description:
- Input Type: gen_text_input
- Field Type: text
- Catalog Type: null

Implementation

Use the userAPIGetCatalogInputDefinition API and pass the advanced catalog name associated with a workflow to view the workflow input fields.

See Also

- Retrieving Output Fields of a Workflow Associated with a Catalog, on page 94
- Retrieving Workflow Input Fields, on page 95
- Retrieving Workflow Output Fields, on page 97

Retrieving Output Fields of a Workflow Associated with a Catalog

Objective

Retrieve the output fields of a workflow that is associated with an advanced catalog. You can view the output fields such as workflow output label, name, description, and type of a workflow.

Context

To know the output fields of the workflow associated with the advanced catalog.

Prerequisites

Catalog must be associated with a workflow.

REST URL

Request

```
/app/api/rest?formatType=json&opName=catalog:userAPIGetCatalogOutputDefinition&opData=
{param0:{"catalogName":"AdvCatalog"}}
```

Response

```
{
"serviceResult":{"workflowOutputFieldList":[{"outputFieldLabel":”name”,"outputFieldName":
"output_0_output1534","outputFieldType":"gen_text_input","outputFieldDescription":""}]},
"serviceError":null, "serviceName":"InfraMgr",
"opName":"catalog:userAPIGetCatalogOutputDefinition" }
```

Components

catalogName—Name of the advanced catalog.

Code

```
public class GetCatalogOutput
{
  public static void main(String[] args) throws Exception
{
  CuicServer api = CuicServer.getAPI("172.29.110.241", "3E17CFFBA7A64C71B8958F40DE2EC9B3",
 "http", 80);
  UserAPICatalog catalog = new UserAPICatalog(api);
  APICatalogParams params = new APICatalogParams();
  params.setCatalogName("AdvCatalog");
  APIWorkflowOutputFieldDefinitionList def =
catalog.userAPIGetCatalogOutputDefinition(params);
  List<APIWorkflowOutputFieldDefinition> list;
  if(def != null){
   list = def.getWorkflowOutputFieldList();
  }else{
   throw new Exception("No output defination found!");
  }
    if(list != null && list.size() > 0){
      for(int i=0;i< list.size();i++){
       System.out.println("Field label: "+list.get(i).getOutputFieldLabel());
```

```
             System.out.println("Field name: "+list.get(i).getOutputFieldName());
            System.out.println("Field description: "+list.get(i).getOutputFieldDescription());

             System.out.println("Field type: "+list.get(i).getOutputFieldType());
            }
        }else{
         throw new Exception("No workflow output field found!");
        }
        //System.out.println("lisst = " + list.getWorkflowOutputFieldList().get(0));
        }
      }
```

Results

The output fields of the workflow associated with the catalog are listed.

Sample Result:

- Field label: output1
- Field name: output_0_output1534
- Field description:
- Field type: gen_text_input

Implementation

Use the userAPIGetCatalogOutputDefinition API and pass the advanced catalog name associated with a workflow to view the workflow output fields.

See Also

- Retrieving Input Fields of a Workflow Associated with a Catalog, on page 92
- Retrieving Workflow Input Fields, on page 95
- Retrieving Workflow Output Fields, on page 97

Retrieving Workflow Input Fields

Objective

Retrieve the input fields of a workflow. You can view the input fields of the workflow such as input label, name, description, type, and isAdmin input type of a workflow.

Context

To know the input fields of the workflow.

Prerequisites

The workflow must be present in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPIGetWorkflowInputs&opData=
{param0:"Expand VSAN Cluster"}
```

Response

```
{ "serviceResult":{"details":[{"name":"input_6_VSAN_Cluster915","label":"VSAN Cluster",
"description":"Select VSAN
Cluster","type":"vsanCluster","catalogType":null,"isOptional":false,
"inputFieldType":"popup-table","isMultiSelect":false,"inputFieldValidator":null,
"isAdminInput":false},{"name":"input_3_Host_Nodes50","label":"Host
Nodes","description":"Host
 Nodes : Ex. 172.29.195.75,172.29.195.76,172.29.195.77","type":"gen_text_input",
"catalogType":null,"isOptional":false,"inputFieldType":"text","isMultiSelect":false,
"inputFieldValidator":null,"isAdminInput":false},{"name":"input_2_Host_User_ID924",
"label":"Host User ID","description":"Enter User
ID","type":"gen_text_input","catalogType":null,
"isOptional":false,"inputFieldType":"text","isMultiSelect":false,"inputFieldValidator":null,
"isAdminInput":false},{"name":"input_3_Host_Password766","label":"Host
Password","description":
"Enter Host Password","type":"password","catalogType":null,"isOptional":false,
"inputFieldType":"password","isMultiSelect":false,"inputFieldValidator":null,
"isAdminInput":false},{"name":"input_6_Host_License325","label":"Host
License","description":"",
"type":"gen_text_input","catalogType":null,"isOptional":true,"inputFieldType":"text",
"isMultiSelect":false,"inputFieldValidator":null,"isAdminInput":false},
{"name":"input_5_DVSwitch176","label":"DVSwitch","description":"Select DVSwitch","type":
"VMwareDVSwitchIdentity","catalogType":null,"isOptional":false,"inputFieldType":"table",
"isMultiSelect":false,"inputFieldValidator":null,"isAdminInput":false},
{"name":"input_9_DVSwitch_Uplink_Portgroup885","label":"DVSwitch Uplink Portgroup",
"description":"Select Uplink Portgroup,According to
DVSwitch","type":"uplinkPortGroupLovList",
"catalogType":null,"isOptional":false,"inputFieldType":"embedded-lov",
"isMultiSelect":false,"inputFieldValidator":null,"isAdminInput":false},
{"name":"input_10_Virtual_SAN_Portgroup153","label":"Virtual SAN Portgroup",
"description":"Select Virtual SAN Portgroup","type":"VMwareDVPortgroupIdentity",
"catalogType":null,"isOptional":false,"inputFieldType":"table","isMultiSelect":false,
"inputFieldValidator":null,"isAdminInput":false},{"name":"input_11_VSAN_IP_Pool_Policy298",
"label":"VSAN IP Pool Policy","description":"Select VSAN IP Pool Policy","type":
"VMWareIPPoolPolicy","catalogType":null,"isOptional":false,"inputFieldType":"popup-table",
"isMultiSelect":false,"inputFieldValidator":null,"isAdminInput":false},
{"name":"input_11_vMotion_Portgroup289","label":"vMotion Portgroup",
"description":"","type":"VMwareDVPortgroupIdentity","catalogType":null,"isOptional":true,
"inputFieldType":"table","isMultiSelect":false,"inputFieldValidator":null,"isAdminInput":false},
{"name":"input_10_vMotion_IP_Pool_Policy807","label":"vMotion IP Pool
Policy","description":"",
"type":"VMWareIPPoolPolicy","catalogType":null,"isOptional":true,"inputFieldType":"popup-table",
"isMultiSelect":false,"inputFieldValidator":null,"isAdminInput":false},
{"name":"input_7_MTU_Size697","label":"MTU Size","description":"Enter MTU Size,
Ex : 1500 or 9000","type":"gen_text_input","catalogType":null,"isOptional":false,
"inputFieldType":"text","isMultiSelect":false,"inputFieldValidator":null,"isAdminInput":false}]},

 "serviceError":null, "serviceName":"InfraMgr","opName":"userAPIGetWorkflowInputs" }
```

Components

param0—Name of the workflow.

Code

```
public class GetWorkflowInputs
{
 public static void main(String[] args) throws Exception
{
  CuicServer api = CuicServer.getAPI("10.23.210.42", "3E17CFFBA7A64C71B8958F40DE2EC9B3",
 "http", 80);
  UserAPIGlobal instance = new UserAPIGlobal(api);
       APIWorkflowInputDetails details = instance.userAPIGetWorkflowInputs("Test1");
       List<APIWorkflowInputDetail> list = null;
    if(details != null){
     list = details.getDetails();
```

```
            }else{
             throw new Exception("No workflow input defination found!");
            }
               if(list != null && list.size() > 0){
                for(int i = 0;i < list.size();i++){
                System.out.println("Field Name: "+list.get(i).getName());
                System.out.println("Field Label: "+list.get(i).getLabel());
                System.out.println("Field Description: "+list.get(i).getDescription());
                System.out.println("Input Type: "+list.get(i).getType());
                System.out.println("Field Type: "+list.get(i).getInputFieldType());
                System.out.println("Is Admin Input Type :"+list.get(i).isAdminInput());
               }
               }else{
                System.out.println("No workflow input found!");
               }
         }
        }
```

Results

The input fields of the workflow are listed.

Sample Result:

- Field Name: input_0_name250
- Field Label: name
- Field Description: person name
- Input Type: gen_text_input
- Field Type: text
- Is Admin Input Type : false

Implementation

Use the userAPIGetWorkflowInputs API and pass the workflow name to view the workflow input fields.

See Also

- Retrieving Input Fields of a Workflow Associated with a Catalog, on page 92
- Retrieving Output Fields of a Workflow Associated with a Catalog, on page 94
- Retrieving Workflow Output Fields, on page 97

Retrieving Workflow Output Fields

Objective

Retrieve the output fields of a workflow. You can retrieve the output fields such as output label, name, description, and type of a workflow.

Context

To know the output fields of the workflow.

Prerequisites

None

REST URL

Request

```
/app/api/rest?formatType=json&opName=
workflow:userAPIGetWorkflowOutputDefinition&opData={param0:"Test1"}
```

Response

```
{ "serviceResult":{"workflowOutputFieldList":[{"outputFieldLabel":"OUTPUT_NAME",
"outputFieldName":"output_0_OUTPUT_NAME75","outputFieldType":"gen_text_input",
"outputFieldDescription":"person's name"}]}, "serviceError":null,
"serviceName":"InfraMgr",
 "opName":"workflow:userAPIGetWorkflowOutputDefinition" }
```

Components

param0—Name of the workflow for which you want to view the output fields.

Code

```
public class GetWorkflowOutputDef
{
 public static void main(String[] args) throws Exception
 {
  CuicServer api = CuicServer.getAPI("10.23.210.42", "3E17CFFBA7A64C71B8958F40DE2EC9B3",
 "http", 80);
  UserAPIWorkflow instance = new UserAPIWorkflow(api);
  APIWorkflowOutputFieldDefinitionList def =
instance.userAPIGetWorkflowOutputDefinition("Test1");
  List<APIWorkflowOutputFieldDefinition> list =null;
  if(def != null){
   list = def.getWorkflowOutputFieldList();
  }else{
   throw new Exception("No workflow output defination found!");
  }
  if(list != null && list.size() > 0){
   for(int i=0;i<list.size();i++){
    System.out.println("Field Name: "+list.get(i).getOutputFieldName());
    System.out.println("Field Label: "+list.get(i).getOutputFieldLabel());
    System.out.println("Field Description: "+list.get(i).getOutputFieldDescription());

    System.out.println("Field Type: "+list.get(i).getOutputFieldType());
   }
  }else{
   System.out.println("No workflow output field found!");
  }
 }
}
```

Results

The output fields of the workflow are listed.

Sample Result:

- Field Name: output_0_OUTPUT_NAME75
- Field Label: OUTPUT_NAME
- Field Description: person's name
- Field Type: gen_text_input

Implementation

Use the userAPIGetWorkflowOutputDefinition API and pass the workflow name to view the workflow output fields.

See Also

- Retrieving Input Fields of a Workflow Associated with a Catalog, on page 92
- Retrieving Output Fields of a Workflow Associated with a Catalog, on page 94
- Retrieving Workflow Input Fields, on page 95

Managing MSP

Toggling MSP Mode

Objective

Enable or disable Managed Service Provider (MSP) mode in Cisco UCS Director.

Context

The MSP mode must be enabled to access MSP users or MSP organization in Cisco UCS Director.

Prerequisites

None

Post Requisites

Restart the Cisco UCS Director services.

REST URL

```
/app/api/rest?formatType=json&opName=auth:userAPIToggleMspMode&opData={param0:
{"action":true,"tenantName":"sample","orgName":"sample"}}
```

Components

The parameters of the userAPIToggleMspMode API are:

- action—Set the value as true to enable the MSP mode. Set the value as false to disable the MSP mode.
- tenantName—The name of the MSP user. This parameter is optional for the MSP disable operation.
- orgName—The name of the MSP organization. This parameter is optional for the MSP disable operation.

Code

```
public static void main(String[] args) throws Exception
{
  CuicServer api = CuicServer.getAPI("172.29.110.241", "3E17CFFBA7A64C71B8958F40DE2EC9B3",
 "http", 80);
  UserAPIAuthConfig instance = new UserAPIAuthConfig(api);
  APIMSPModeParams params = new APIMSPModeParams();
  params.setAction(true);
  params.setTenantName("MSP Users");
```

```
   params.setOrgName("MSP Organization");
   String result = instance.userAPIToggleMspMode(params);
   System.out.println("Result: "+result);
  }
```

Results

The status message is displayed according to the set MSP mode.

- If the MSP mode is enabled, the following message is displayed:

```
MSP Mode is enabled successfully. Restart the Cisco UCS Director services to reflect
 the changes.
```

- If the MSP mode is disabled, the following message is displayed:

```
MSP Mode is disabled successfully. Restart the Cisco UCS Director services to reflect
 the changes.
```

Implementation

Pass the MSP user name and MSP organization name, and set the action as true in the userAPIToggleMspMode API to enable the MSP mode.

Managing Data Stores

Retrieving an Eligible List of Data Store Clusters

Objective

Retrieve a list of eligible data store clusters that can be used to add a new disk to an existing VM.

Context

When a new disk is added to an existing VM, you have to provide a valid and available data store cluster. You can use the userAPIGetEligibleDataStoreClustersForCreateNewDisk API to get the list of eligible data store clusters that can be used to provide a valid data store cluster during new disk addition. This API returns the list of eligible data store cluster based on the vmId, diskSize, and diskType parameters.

Prerequisites

The vmId must refer to an existing VM in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=genericvm:
userAPIGetEligibleDataStoreClustersForCreateNewDisk&opData={param0:{"vmId":86,
"diskSize":10.0,"diskType":0}}
```

Response

```
{"serviceResult":{"name":"DS_Cluster_2","capacity":1099511627776,"freespace":26096259072,
"drsEnabled":"Enabled","ioMetrics":"Enabled","automationLevel":"automated","utilitySpaceThreshold":50,
"ioLatencyThreshold":5,"datastores":"QA_DS01","type":"NFS","accountName":"VMC001",
"vmwareDatacenterName":"New Datacenter"}], "serviceError":null, "serviceName":"InfraMgr",

"opName":"genericvm:userAPIGetEligibleDataStoreClustersForCreateNewDisk" }
```

Components

- int vmId—The vm Id of an existing VM in Cisco UCS Director.
- double diskSize—The size of disk that needs to be added, in GB.
- int diskType—The type of the disk that needs to be added. The valid disk types are:
    - 0—System
    - 1—Data
    - 2—Database
    - 3—Log
    - 4—Swap

Code

```
import java.util.List;
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.UserAPIVMware;
import com.cisco.cuic.api.models.vmware.CreateVMDiskParams;
import com.cisco.cuic.api.models.vmware.VMWareDatastoreCluster;

public class TestUserAPIGetEligibleDataStoreClustersForCreateNewDisk {
 public static void main(String[] args)  {
  CuicServer server = CuicServer.getAPI("172.22.234.172",
"91E7A134048F45FD88A29FCBBFD2C233", "https", 443, 12000, "none", "");
  UserAPIVMware instance = new UserAPIVMware(server);
  CreateVMDiskParams params = new CreateVMDiskParams();
  params.setVmId(86);
  params.setDiskSize(10.0);
  params.setDiskType(0);
  List<VMWareDatastoreCluster> vmwareDatastoreClusterList;
  try {
   vmwareDatastoreClusterList =
instance.userAPIGetEligibleDataStoreClustersForCreateNewDisk(params);
   System.out.println("list size = "+vmwareDatastoreClusterList.size());
   for(VMWareDatastoreCluster vmwareDatastoreCluster:vmwareDatastoreClusterList){
    System.out.println(vmwareDatastoreCluster.getName());
   }
  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }
 }
}
```

Results

The data store clusters list is returned based on the passed vmId, diskSize, and diskType parameters.

Implementation

Set the parameters mentioned in the component section to get the list of available data store clusters.

See Also

Retrieving an Eligible List of Data Stores

Retrieving an Eligible List of Data Stores

Objective

Retrieve a list of eligible data stores that can be used to add a new disk to an existing VM.

Context

When a new disk is added to an existing VM, you have to provide a valid and available data store. You can use the userAPIGetEligibleDataStoresForCreateNewDisk API to get the list of eligible data stores that can be used to provide a valid data store during new disk addition. This API returns the list of eligible data store based on the vmId, diskSize, and diskType parameters.

Prerequisites

The vmId must refer to an existing VM in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=
genericvm:userAPIGetEligibleDataStoresForCreateNewDisk&opData={param0:{"vmId":86,
"diskSize":20.0,"diskType":0}}
```

Response

```
{ "serviceResult":[{"hostName":"172.29.109.65","accountName":"VMC001","name":
"datastore1 (2)","capacityBytes":993211187200,"freeBytes":37044092928,"type":"VMFS",
"url":"ds:///vmfs/volumes/51b73895-bfedae18-1852-4c4e353d287e/","isMultiHost":false,
"uncomitted":515370287104,"vmCount":26,"numHosts":1,"mountAccessMode":"readWrite",
"mountPath":"/vmfs/volumes/51b73895-bfedae18-1852-4c4e353d287e","vmfsBlockSizeMB":1,
"vmfsMaxBlocks":63963136,"vmfsVersion":"5.58","luns":[{"uuid":null,"canonicalName":
"naa.600605b005a7ee201949f39808290917","lunType":"disk","scsiLevel":5,"vendor":"LSI",
"key":"key-vim.host.ScsiDisk-0200000000600605b005a7ee201949f398082909174d5239323731",
"deviceType":"disk","displayName":"Local LSI Disk (naa.600605b005a7ee201949f39808290917)",
"model":"MR9271-8i","deviceName":"/vmfs/devices/disks/naa.600605b005a7ee201949f39808290917",
"hostName":"10.10.109.65","accountName":"VMC001","datacenterName":"New Datacenter",
"datastoreName":"datastore1 (2)"}],"nfsRemoteHost":null,"nfsRemotePath":null,
"nfsRemoteUsername":null,"vmwareDatacenterName":"New Datacenter","adapterLunMap":[],
"vmList":null,"accessible":true,"localfsRemotePath":null}], "serviceError":null,
"serviceName":"InfraMgr",
"opName":"genericvm:userAPIGetEligibleDataStoresForCreateNewDisk" }
```

Components

- int vmId—The vm Id of an existing VM in Cisco UCS Director.
- double diskSize—The size of disk that needs to be added, in GB.
- int diskType—The type of the disk that needs to be added. The valid disk types are:
    - 0—System
    - 1—Data
    - 2—Database
    - 3—Log
    - 4—Swap

Code

```
import java.util.List;
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.UserAPIVMware;
import com.cisco.cuic.api.models.vmware.CreateVMDiskParams;
import com.cisco.cuic.api.models.vmware.DataStoreInfo;

public class TestUserAPIGetEligibleDataStoresForCreateNewDisk {
 public static void main(String[] args) {
  CuicServer server = CuicServer.getAPI("172.22.234.172",
"91E7A134048F45FD88A29FCBBFD2C233", "https", 443, 12000, "none", "");
  UserAPIVMware instance = new UserAPIVMware(server);
  CreateVMDiskParams params = new CreateVMDiskParams();
  params.setVmId(86);
  params.setDiskSize(10.0);
  params.setDiskType(0);
  List<DataStoreInfo> dataStoreInfoList;
  try {
   dataStoreInfoList = instance.userAPIGetEligibleDataStoresForCreateNewDisk(params);
   for(DataStoreInfo datastoreInfo: dataStoreInfoList){
    System.out.println("Data store name: "+datastoreInfo.getName());
   }
  } catch (Exception e) {

   e.printStackTrace();
  }
 }
}
```

Results

The data stores list is returned based on the passed vmId, diskSize, and diskType parameters.

Implementation

Set the parameters mentioned in the component section to get the list of available data stores.

See Also

Retrieving an Eligible List of Data Store Clusters

Managing Reports

Viewing Available Reports Definition

Objective

You can use the userAPIGetAvailableReports API to view the type, ID, and label of all the reports that are available in Cisco UCS Director for the given contextName and contextValue.

Context

To view the definition of reports that are available for the given contextName and contextValue.

Prerequisites

The given contextName and contextValue must be available in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPIGetAvailableReports&opData
={param0:"22",param1:"Jha_VDC_VMware_82"}
```

Response

```
{
"serviceResult":[{"reportLabel":"Summary","reportId":"SUMMARY-V0","reportType":"tabular"},
{"reportLabel":"Summary","reportId":"SUMMARY-V14","reportType":"tabular"},{"reportLabel":"VMs",
"reportId":"VMS-T0","reportType":"tabular"},{"reportLabel":"Events","reportId":"EVENTS-T0",
"reportType":"tabular"},{"reportLabel":"Static IP Assignment",
"reportId":"STATIC-IP-ASSIGNMENT-T0","reportType":"tabular"},{"reportLabel":"Deleted
VMs",
"reportId":"DELETED-VMS-T0","reportType":"tabular"},{"reportLabel":"VDC Compliance on
VMs",
"reportId":"VDC-COMPLIANCE-ON-VMS-T0","reportType":"tabular"},{"reportLabel":"Trend:vDC
 CPU Usage",
"reportId":"TREND-VDC-CPU-USAGE-H0","reportType":"trend"},{"reportLabel":"Trend:vDC
Memory Usage",
"reportId":"TREND-VDC-MEMORY-USAGE-H0","reportType":"trend"},{"reportLabel":"Trend:vDC
 Disk Usage",
"reportId":"TREND-VDC-DISK-USAGE-H0","reportType":"trend"},{"reportLabel":"Trend:vDC
Network Usage",
"reportId":"TREND-VDC-NETWORK-USAGE-H0","reportType":"trend"},{"reportLabel":"Trend:Number
 of VMs",
"reportId":"TREND-NUMBER-OF-VMS-H0","reportType":"trend"},{"reportLabel":"Trend:VM
Additions
& Deletions","reportId":"TREND-VM-ADDITIONS-&-DELETIONS-H0","reportType":"trend"},
{"reportLabel":"Trend: vDC CPU Usage (GHz)","reportId":"TREND-VDC-CPU-USAGE-(GHZ)-H14",
"reportType":"trend"},{"reportLabel":"Trend: vDC Memory Usage GB","reportId":
"TREND-VDC-MEMORY-USAGE-GB-H14","reportType":"trend"},{"reportLabel":"Trend: vDC Disk
Usage",
"reportId":"TREND-VDC-DISK-USAGE-H14","reportType":"trend"},{"reportLabel":"Trend: vDC
 Network Usage",
"reportId":"TREND-VDC-NETWORK-USAGE-H14","reportType":"trend"},
{"reportLabel":"Trend: Number of
VMs","reportId":"TREND-NUMBER-OF-VMS-H14","reportType":"trend"},
{"reportLabel":"Trend: VM Additions &
Deletions","reportId":"TREND-VM-ADDITIONS-&-DELETIONS-H14",
"reportType":"trend"},{"reportLabel":"Trend: Total Cost","reportId":"TREND-TOTAL-COST-H0",
"reportType":"trend"},{"reportLabel":"Trend: VM Cost","reportId":"TREND-VM-COST-H0",
"reportType":"trend"},{"reportLabel":"Trend: CPU Cost","reportId":"TREND-CPU-COST-H0",
"reportType":"trend"},{"reportLabel":"Trend: Memory
Cost","reportId":"TREND-MEMORY-COST-H0",
"reportType":"trend"},{"reportLabel":"Trend: Network
Cost","reportId":"TREND-NETWORK-COST-H0",
"reportType":"trend"},{"reportLabel":"Trend: Total
Cost","reportId":"TREND-TOTAL-COST-H14",
"reportType":"trend"},{"reportLabel":"Trend: VM Cost","reportId":"TREND-VM-COST-H14",
"reportType":"trend"},{"reportLabel":"Trend: CPU Cost","reportId":"TREND-CPU-COST-H14",
"reportType":"trend"},{"reportLabel":"Trend: Memory
Cost","reportId":"TREND-MEMORY-COST-H14",
"reportType":"trend"},{"reportLabel":"Trend: Network
Cost","reportId":"TREND-NETWORK-COST-H14",
"reportType":"trend"},{"reportLabel":"Trend:Snapshot File
Size","reportId":"TREND-SNAPSHOT-FILE-SIZE-H0",
"reportType":"trend"},{"reportLabel":"VMs","reportId":"VMS-T14","reportType":"tabular"},
{"reportLabel":"Events","reportId":"EVENTS-T14","reportType":"tabular"},{"reportLabel":"Deleted
 VMs",
"reportId":"DELETED-VMS-T14","reportType":"tabular"},{"reportLabel":"Chargeback",
"reportId":"CHARGEBACK-T12","reportType":"tabular"},{"reportLabel":"Resource Accounting",
"reportId":"RESOURCE-ACCOUNTING-T12","reportType":"tabular"},{"reportLabel":"Resource
Accounting Details",
"reportId":"RESOURCE-ACCOUNTING-DETAILS-T12","reportType":"tabular"},
{"reportLabel":"CPU Usage (MHz)","reportId":"CPU-USAGE-(MHZ)-S0","reportType":"snapshot"},
{"reportLabel":"Memory Usage
(GB)","reportId":"MEMORY-USAGE-(GB)-S0","reportType":"snapshot"},
{"reportLabel":"Disk Usage (GB)","reportId":"DISK-USAGE-(GB)-S0","reportType":"snapshot"},
{"reportLabel":"Number of Events by
Severity","reportId":"NUMBER-OF-EVENTS-BY-SEVERITY-S0",
"reportType":"snapshot"},{"reportLabel":"CPU Usage
(MHz)","reportId":"CPU-USAGE-(MHZ)-S14",
"reportType":"snapshot"},{"reportLabel":"Memory Usage
(GB)","reportId":"MEMORY-USAGE-(GB)-S14",
"reportType":"snapshot"},{"reportLabel":"Disk Usage
(GB)","reportId":"DISK-USAGE-(GB)-S14",
"reportType":"snapshot"}], "serviceError":null, "serviceName":"InfraMgr",
"opName":"userAPIGetAvailableReports" }
```

Components

The parameters of the userAPIGetAvailableReports API are:

- String param0—The name of the report context.
- int param1—The value of the report context.

For more information on the name and value of the report context, see the Appendix B: Report Context Types and Report Context Names appendix in the Cisco UCS Director Open Automation Cookbook.

Code

```
import java.util.List;
import com.cisco.cuic.api.models.UserAPIGlobal;
import com.cisco.cuic.api.client.APIReportDefinition;
import com.cisco.cuic.api.client.CuicServer;

public class TestuserAPIGetAvailableReports {

 public static void main(String[] args) throws Exception {
   CuicServer server =
CuicServer.getAPI("172.29.110.222","96408900345D40C0BC889E4F41C2E094", "https", 443);
   UserAPIGlobal instance = new UserAPIGlobal(server);
   List<APIReportDefinition>
apiReportDefinitionList=instance.userAPIGetAvailableReports("22", "Jha_VDC_VMware-82");

   int i=0;
   for(APIReportDefinition apiReportDefinition: apiReportDefinitionList){
    System.out.println("Report_"+i);
    System.out.println("reportLabel: "+apiReportDefinition.getReportLabel());
    System.out.println("reportID: "+apiReportDefinition.getReportId());
    System.out.println("reportType: "+apiReportDefinition.getReportType());
    i++;
   }
 }
}
```

Results

The type, ID, and label of reports that are available for the given contextName and contextValue are displayed.

Implementation

Use the userAPIGetAvailableReports API and pass the contextName and contextValue to view the list of API report definitions for the given contextName and contextValue.

See Also

Viewing Historical Report

Viewing Resource Usage Report

Viewing Snapshot Report

Viewing Tabular Reports

Viewing Historical Report

Objective

You can use the userAPIGetHistoricalReport API to view the historical data of a report context, such as VDC CPU usage trend report, for a specific time period. The report context refers to contextName, contextValue, and reportId.

Context

To view the historical data of a report context for a specific time period.

Prerequisites

The given contextName and contextValue must be available in Cisco UCS Director. The time period must be one of the specified durations in Cisco UCS Director or the custom time period in the format supported by Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPIGetHistoricalReport&opData=
{param0:"22",param1:"2",param2:"TREND-VDC-CPU-USAGE-H0",param3:"hourly"}
```

Response

```
{ "serviceResult":{"series":[{"paramName":"vdc.cpu.usage.average:gigaHertz",
"paramLabel":"CPU Usage (GHz)","values":[{"timestamp":1467594752222,"min":0.0,"max":0.0,
"avg":0.165},{"timestamp":1467598352222,"min":0.0,"max":0.0,"avg":0.165}],"precision":2,"units":""}]},

 "serviceError":null, "serviceName":"InfraMgr", "opName":"userAPIGetHistoricalReport"
}
```

Components

The parameters of the userAPIGetHistoricalReport API are:

- String param0—The name of the report context.
- int param1—The value of the report context.
- int param2—The unique identifier of the report.
- int Param3—The time duration for which you want to generate the historical report. The predefined valid time durations are hourly, daily, weekly, and monthly. The custom time duration must start with the prefix “custom:”.

For more information on the name and value of the report context, see the Appendix B: Report Context Types and Report Context Names in the Cisco UCS Director Open Automation Cookbook.

Code

```
import java.util.List;
import com.cisco.cuic.api.client.APIHistoricalReport;
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.client.DataSample;
import com.cisco.cuic.api.client.HistoricalDataSeries;
import com.cisco.cuic.api.models.UserAPIGlobal;

public class TestuserAPIGetHistoricalReport {

 public static void main(String[] args) throws Exception {
  CuicServer server =
CuicServer.getAPI("172.29.110.222","96408900345D40C0BC889E4F41C2E094", "https", 443);
  UserAPIGlobal instance = new UserAPIGlobal(server);
  APIHistoricalReport historicalReport=instance.userAPIGetHistoricalReport("22", "2",
"TREND-VDC-CPU-USAGE-H0", "hourly");
  List<HistoricalDataSeries> historicalDataSeriesList = historicalReport.getSeries();
  int i=0;
  for(HistoricalDataSeries hds: historicalDataSeriesList){
   System.out.println("****Series_"+i+"****");
   System.out.println("paramName: "+hds.getParamName());
   System.out.println("paramLabel: "+hds.getParamLabel());
   DataSample[] dataSampleArr=hds.getValues();
   for(DataSample ds:dataSampleArr ){
    System.out.println("timestamp: "+ds.getTimestamp());
    System.out.println("min: "+ds.getMin());
    System.out.println("max: "+ds.getMax());
    System.out.println("avg: "+ ds.getAvg());
   }
   System.out.println("precision: "+hds.getPrecision());
   System.out.println("units: "+hds.getUnits());
   i++;
  }
 }
}
```

Results

The historical data of a report context for a specific time period is displayed.

Implementation

Use the userAPIGetHistoricalReport API and pass the contextName, contextValue, and time duration to view the historical data of a report context for a specific time period.

See Also

Viewing Available Reports Definition

Viewing Resource Usage Report

Viewing Snapshot Report

Viewing Tabular Reports

Viewing Resource Usage Report

Objective

The report of resources used in the cloud infrastructure management can be viewed using the userAPIGetResourceUsageCostSummary API. Using this API, administrator and end user can view the daily, weekly, and monthly resource usage report.

Context

To view the infrastructure utilization or cost reporting of resources on daily, weekly, or monthly basis.

Prerequisites

None

REST URL

This section provides the REST URL for viewing resource usage report in different scenarios:

Example 1: Daily usage report of a virtual machine (VM) without resourceName

Request

```
/app/api/rest?formatType=json&opName=
chargeback:userAPIGetResourceUsageCostSummary&opData={param0:{"requestParam":
[{"name":"vmid","value":"56"}],"fromTimeInMilliSeconds":1442946600000,
"toTimeInMilliSeconds":1443032999000}}
```

Response

```
{ "serviceResult":{"resourceType":"VM","duration":"Daily_22_9_2015",
"responseParamList":{"list":[{"name":"vmid","value":"56"}]},
"resourceUsageCostSummary":[{"groupId":1,"groupName":"Default Group","vdcId":0,
"onetimeCost":8.0,"activeCost":56.0,"inactiveCost":4.0,
"applicationCost":23.0,"totalCost":5.0,"fixedCost":4.0,"catalogItemName":
"SDK","cpuResourceUsageCost":{"cpu_GHz_Hours":111.99998388000004,
"cpuCost":4.0,"reservedCpuGhzCost":9.0,"usedCpuGhzCost":65.0,"reservedCpuGhz":0.0,
"usedCpuGhz":0.0,"cpuCores":40,"cpuCoreCost":56.0}],"diskResourceUsageCost":
[{"committedDiskGB":40.60000000000001,"uncommittedDiskGB":1.4000000000000008,
"committedDiskGBCost":56.0,"uncommittedDiskGBCost":567.0}],"memoryResourceUsageCost":
[{"memory":67.0,"memoryCost":67.0,"reservedMemoryGBCost":7.0,"usedMemoryGBCost":12.0,
"usedMemoryGB":0.0,"reservedMemoryGB":0.0}],"networkResourceUsageCost":
[{"netRxUsageGB":0.0,"netTxUsageGB":76.0,"netRxUsageGBCost":0.00006389617919921875,
"netTxUsageGBCost":0.00008296966552734375}],"physicalServerResourceUsageCost":
[{"halfLengthBlade":false,"bladeType":"","fullBladeCost":67.0,"halfBladeCost":76.0,
"serverCost":5.0}]}]}, "serviceError":null, "serviceName":"InfraMgr",
"opName":"chargeback:userAPIGetResourceUsageCostSummary" }
```

Example 2: Daily usage report of a group without resourceName

Request

```
/app/api/rest?formatType=json&opName=
chargeback:userAPIGetResourceUsageCostSummary&opData={param0:{"requestParam":
[{"name":"Group","value":"Default Group"}],"fromTimeInMilliSeconds":1442946600000,
"toTimeInMilliSeconds":1443032999000}}
```

Response

```
{ "serviceResult":{"resourceType":null,"duration":"Daily_22_9_2015",
"responseParamList":{"list":[{"name":"Group","value":"Default Group"}]},
"resourceUsageCostSummary":[{"groupId":1,"groupName":"Default Group","vdcId":0,
"onetimeCost":8.0,"activeCost":56.0,"inactiveCost":4.0,"applicationCost":23.0,
"totalCost":5.0,"fixedCost":4.0,"catalogItemName":"SDK","cpuResourceUsageCost":
[{"cpu_GHz_Hours":351.99997083999995,"cpuCost":4.0,"reservedCpuGhzCost":9.0,
"usedCpuGhzCost":65.0,"reservedCpuGhz":0.0,"usedCpuGhz":0.0,"cpuCores":120,
"cpuCoreCost":56.0}],"diskResourceUsageCost":[{"committedDiskGB":1225.1200000000001,
"uncommittedDiskGB":3974.999999999999,"committedDiskGBCost":56.0,
"uncommittedDiskGBCost":567.0}],"memoryResourceUsageCost":[{"memory":240.0,
"memoryCost":67.0,"reservedMemoryGBCost":7.0,"usedMemoryGBCost":12.0,
"usedMemoryGB":0.0,"reservedMemoryGB":0.0}],"networkResourceUsageCost":
[{"netRxUsageGB":0.0,"netTxUsageGB":76.0,"netRxUsageGBCost":0.00006389617919921875,
"netTxUsageGBCost":0.00008296966552734375}],"physicalServerResourceUsageCost":
[{"halfLengthBlade":false,"bladeType":"","fullBladeCost":67.0,"halfBladeCost":76.0,
"serverCost":5.0}]}]}, "serviceError":null, "serviceName":"InfraMgr",
"opName":"chargeback:userAPIGetResourceUsageCostSummary" }
```

Example 3: Daily usage report of a virtual data center (VDC) without resourceName

Request

```
/app/api/rest?formatType=json&opName=
chargeback:userAPIGetResourceUsageCostSummary&opData={param0:
{"requestParam":[{"name":"vdcid","value":"1"}],"fromTimeInMilliSeconds":1442946600000,
"toTimeInMilliSeconds":1443032999000}}
```

Response

```
{ "serviceResult":{"resourceType":null,"duration":"Daily_22_9_2015",
"responseParamList":{"list":[{"name":"vdcName","value":"Default vDC"}]},
"resourceUsageCostSummary":[{"groupId":1,"groupName":"Default Group","vdcId":0,
"onetimeCost":8.0,"activeCost":56.0,"inactiveCost":4.0,"applicationCost":23.0,
"totalCost":5.0,"fixedCost":4.0,"catalogItemName":"SDK","cpuResourceUsageCost":
[{"cpu_GHz_Hours":351.99997083999995,"cpuCost":4.0,"reservedCpuGhzCost":9.0,
"usedCpuGhzCost":65.0,"reservedCpuGhz":0.0,"usedCpuGhz":0.0,"cpuCores":120,
"cpuCoreCost":56.0}],"diskResourceUsageCost":[{"committedDiskGB":1225.1200000000001,
"uncommittedDiskGB":3974.999999999999,"committedDiskGBCost":56.0,
"uncommittedDiskGBCost":567.0}],"memoryResourceUsageCost":[{"memory":240.0,
"memoryCost":67.0,"reservedMemoryGBCost":7.0,"usedMemoryGBCost":12.0,
"usedMemoryGB":0.0,"reservedMemoryGB":0.0}],"networkResourceUsageCost":
[{"netRxUsageGB":0.0,"netTxUsageGB":76.0,"netRxUsageGBCost":0.00006389617919921875,
"netTxUsageGBCost":0.00008296966552734375}],"physicalServerResourceUsageCost":
[{"halfLengthBlade":false,"bladeType":"","fullBladeCost":67.0,"halfBladeCost":76.0,
"serverCost":5.0}]}]}, "serviceError":null, "serviceName":"InfraMgr",
"opName":"chargeback:userAPIGetResourceUsageCostSummary" }
```

Example 4: Daily usage report of a CPU in a VM

Request

```
/app/api/rest?formatType=json&opName=
chargeback:userAPIGetResourceUsageCostSummary&opData={param0:{"requestParam":
[{"name":"vmid","value":"56"},{"name":"resourceName","value":"CPU"}],
"fromTimeInMilliSeconds":1442946600000,"toTimeInMilliSeconds":1443032999000}}
```

Response

```
{ "serviceResult":{"resourceType":"VM","duration":"Daily_22_9_2015",
"responseParamList":{"list":[{"name":"vmid","value":"56"}]},
"resourceUsageCostSummary":[{"groupId":1,"groupName":"Default Group","vdcId":0,
"onetimeCost":8.0,"activeCost":56.0,"inactiveCost":4.0,
"applicationCost":23.0,"totalCost":5.0,"fixedCost":4.0,"catalogItemName":
"SDK","cpuResourceUsageCost":{"cpu_GHz_Hours":111.99998388000004,"cpuCost":4.0,
"reservedCpuGhzCost":9.0,"usedCpuGhzCost":2.6,"reservedCpuGhz":0.0,"usedCpuGhz":0.0,
"cpuCores":40,"cpuCoreCost":56.0}],"diskResourceUsageCost":null,
"memoryResourceUsageCost":null,"networkResourceUsageCost":null,
"physicalServerResourceUsageCost":null}]}, "serviceError":null,
"serviceName":"InfraMgr", "opName":"chargeback:userAPIGetResourceUsageCostSummary" }
```

Example 5: Daily usage report of a disk in a group

Request

```
/app/api/rest?formatType=json&opName=
chargeback:userAPIGetResourceUsageCostSummary&opData={param0:
{"requestParam":[{"name":"Group","value":"Default Group"},
{"name":"resourceName","value":"disk"}],"fromTimeInMilliSeconds":1442946600000,
"toTimeInMilliSeconds":1443032999000}}
```

Response

```
{ "serviceResult":{"resourceType":null,"duration":"Daily_22_9_2015",
"responseParamList":{"list":[{"name":"Group","value":"Default Group"}]},
"resourceUsageCostSummary":[{"groupId":1,"groupName":"Default Group","vdcId":0,
"onetimeCost":8.0,"activeCost":56.0,"inactiveCost":4.0,"applicationCost":23.0,
"totalCost":5.0,"fixedCost":4.0,"catalogItemName":"SDK","cpuResourceUsageCost":null,
"diskResourceUsageCost":[{"committedDiskGB":1225.1200000000001,
"uncommittedDiskGB":3974.999999999999,"committedDiskGBCost":56.0,
"uncommittedDiskGBCost":567.0}],"memoryResourceUsageCost":null,
"networkResourceUsageCost":null,"physicalServerResourceUsageCost":null}]},
 "serviceError":null, "serviceName":"InfraMgr",
"opName":"chargeback:userAPIGetResourceUsageCostSummary" }
```

Example 6: Daily usage report of a VM in VDC

Request

```
/app/api/rest?formatType=json&opName=
chargeback:userAPIGetResourceUsageCostSummary&opData={param0:
{"requestParam":[{"name":"vdcid","value":"1"},{"name":"resourceName",
"value":"VM"}],"fromTimeInMilliSeconds":1442946600000,
"toTimeInMilliSeconds":1443032999000}}
```

Response

```
{ "serviceResult":{"resourceType":null,"duration":"Daily_22_9_2015",
"responseParamList":{"list":[{"name":"vdcName","value":"Default vDC"}]},
"resourceUsageCostSummary":[{"groupId":1,"groupName":"Default Group","vdcId":0,
"onetimeCost":8.0,"activeCost":56.0,"inactiveCost":4.0,"applicationCost":23.0,
"totalCost":5.0,"fixedCost":4.0,"catalogItemName":"SDK","cpuResourceUsageCost":
[{"cpu_GHz_Hours":351.99997083999995,"cpuCost":4.0,"reservedCpuGhzCost":9.0,
"usedCpuGhzCost":65.0,"reservedCpuGhz":0.0,"usedCpuGhz":0.0,"cpuCores":0,
"cpuCoreCost":56.0}],"diskResourceUsageCost":[{"committedDiskGB":1225.1200000000001,
"uncommittedDiskGB":3974.999999999999,"committedDiskGBCost":56.0,
"uncommittedDiskGBCost":567.0}],"memoryResourceUsageCost":[{"memory":240.0,
"memoryCost":67.0,"reservedMemoryGBCost":7.0,"usedMemoryGBCost":12.0,
"usedMemoryGB":0.0,"reservedMemoryGB":0.0}],"networkResourceUsageCost":
[{"netRxUsageGB":0.0,"netTxUsageGB":76.0,"netRxUsageGBCost":0.0000638961791992l875,
"netTxUsageGBCost":0.00008296966552734375}],"physicalServerResourceUsageCost":null}]},

 "serviceError":null, "serviceName":"InfraMgr",
"opName":"chargeback:userAPIGetResourceUsageCostSummary" }
```

Components

The parameters of the userAPIGetResourceUsageCostSummary API are:

- APINameValueList requestParam—List of requested report parameters in the name and value pair format. The possible values of name are vmId, vdcName, groupName, and resourceName. If the resourceName is passed as the name, the possible values of resourceName are cpu, disk, memory, VM, BM, network, or empty value (" ").

  The possible combination of parameters for viewing the resource usage report:

Name	**resourceName**
vmid	CPU /Disk/VM/BM/Memory/Network
vdcid	Disk
Group	VM
vmid	None
vdcid	None

Name	resourceName
Group	None

- fromTimeInMilliSeconds—The start time of the report in milliseconds. For example, the date and time string Mon Feb 16 2015 00:00:00 GMT-0400 (Eastern Daylight Time) is represented as 1424059200000 in milliseconds.
- toTimeInMilliSeconds—The end time of the report in milliseconds. For example, the date and time string Sun Feb 22 2015 00:00:00 GMT-0400 (Eastern Daylight Time) is represented as 1424577600000 in milliseconds.

Code

```
public class GetResourceUsageCostSummaryTest {

 public static void main(String[] args)throws Exception {
  CuicServer api = CuicServer.getAPI("172.29.110.128", "3E17CFFBA7A64C71B8958F40DE2EC9B3",
 "http", 80);
  UserAPIChargeBack instance = new UserAPIChargeBack(api);
  APIResourceUsageCostParams costParams = new APIResourceUsageCostParams();
  APINameValue value=new APINameValue();
  value.setName("vmid");
  value.setValue("56");
   List<APINameValue> requestParam=new ArrayList<APINameValue>();

   requestParam.add(value);
   costParams.setRequestParam(requestParam);
   costParams.setFromTimeInMilliSeconds(1442946600000l);
   costParams.setToTimeInMilliSeconds(1443032999000l);


  APIResourceUsageCostSummaryResponse response =
instance.userAPIGetResourceUsageCostSummary(costParams);
  System.out.println("Duration :" +response.getDuration());
  System.out.println("ResourceType :" +response.getResourceType());
  APINameValueList list=response.getResponseParamList();
  List<APINameValue> apivalue= list.getList();
  for(int i=0;i<apivalue.size();i++){
   System.out.println("Name :" + apivalue.get(i).getName());
   System.out.println("Value :" + apivalue.get(i).getValue());
  }

  List<APIResourceUsageCostSummary> summary=response.getResourceUsageCostSummary();
  for(int i=0;i<summary.size();i++){
   System.out.println("catalogItemName :" + summary.get(i).getCatalogItemName());
   //System.out.println("cpuResourceUsageCost" +
summary.get(i).getCpuResourceUsageCost());

   APICPUResourceUsageCost[] cpus=summary.get(i).getCpuResourceUsageCost();
   for(int j=0;j<cpus.length;j++){
   System.out.println("Cpu_GHz_Hours : " + cpus[j].getCpu_GHz_Hours());
   System.out.println("CPU CoreCost : " + cpus[j].getCpuCoreCost());
   System.out.println("Cpu Cores :" + cpus[j].getCpuCores());
   System.out.println("Cpu Cost :" + cpus[j].getCpuCost());
   System.out.println("Reserved CpuGHZCost" + cpus[j].getReservedCpuGhzCost());
   System.out.println("Used Cpu Cost" + cpus[j].getUsedCpuGhzCost());
   }
  }
 }
 }
```

Results

The requested resource usage report is displayed.

Implementation

Use the userAPIGetResourceUsageCostSummary API and pass the report parameter in the name and value format to view the resource usage report.

See Also

Sample code for viewing monthly resource usage report:

```
/app/api/rest?formatType=json&opName=
chargeback:userAPIGetResourceUsageCostSummary&opData={param0:{"requestParam":
[{"name":"vmid","value":"56"},{"name":"resourceName","value":"CPU"}],
"fromTimeInMilliSeconds":1441081800000,"toTimeInMilliSeconds":1443587400000}}
```

Sample code for viewing weekly resource usage report:

```
/app/api/rest?formatType=json&opName=
chargeback:userAPIGetResourceUsageCostSummary&opData={param0:{"requestParam":
[{"name":"vmid","value":"56"},{"name":"resourceName","value":"CPU"}],
"fromTimeInMilliSeconds":1442982600000,"toTimeInMilliSeconds":1443587400000}}
```

The possible combination of inputs for viewing the report are:

- Monthly/weekly/daily timestamp, vmid, and resourceName as CPU, Disk, VM, BM, memory, or network.
- Monthly/weekly/daily timestamp, vdcid, and resourceName as CPU, Disk, VM, BM, memory, or network.
- Monthly/weekly/daily timestamp, Group, and resourceName as CPU, Disk, VM, BM, memory, or network.
- Monthly/weekly/daily timestamp, vmid without any resourceName.
- Monthly/weekly/daily timestamp, vdcid without any resourceName.
- Monthly/weekly/daily timestamp, Group without any resourceName.

Viewing Available Reports Definition

Viewing Historical Report

Viewing Snapshot Report

Viewing Tabular Reports

Viewing Snapshot Report

Objective

You can view the snapshot of a report context using the userAPIGetInstantDataReport API. The report context refers to contextName, contextValue, and reportId.

Context

To view the snapshot of a report context.

Prerequisites

The given contextName, contextValue, and reportId must be available in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPIGetInstantDataReport&opData=
{param0:"1",param1:"Jha_Vmware_Cloud_82",param2:"VMS-ACTIVE-VS-INACTIVE-S0"}
```

Response

```
{ "serviceResult":{"categoryAxisName":null,"valueAxisName":"Active vs Inactive",
"categories":[{"categoryName":"","nameValuePairs":[{"name":"Active VMs","value":"42"},
{"name":"Inactive VMs","value":"29"}]}]}, "serviceError":null, "serviceName":"InfraMgr",

"opName":"userAPIGetInstantDataReport" }
```

Components

The parameters of the userAPIGetInstantDataReport API are:

- String param0—The name of the report context.
- int param1—The value of the report context.
- int param2—The unique identifier of the report.

For more information on the name and value of the report context, see the Appendix B: Report Context Types and Report Context Names in the Cisco UCS Director Open Automation Cookbook.

Code

```
import java.util.List;
import com.cisco.cuic.api.models.UserAPIGlobal;
import com.cisco.cuic.api.client.APISnapshotReport;
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.client.ReportNameValuePair;
import com.cisco.cuic.api.client.SnapshotReportCategory;

public class TestuserAPIGetInstantDataReport {

 public static void main(String[] args) throws Exception {
  CuicServer server =
CuicServer.getAPI("172.29.110.222","96408900345D40C0BC889E4F41C2E094", "https", 443);
  UserAPIGlobal instance = new UserAPIGlobal(server);
  APISnapshotReport apiSanpshotReport=instance.userAPIGetInstantDataReport("1",
"Jha_Vmware_Cloud_82", "VMS-ACTIVE-VS-INACTIVE-S0");
  System.out.println("categoryAxisName: "+apiSanpshotReport.getCategoryAxisName());
  System.out.println("valueAxisName: "+apiSanpshotReport.getValueAxisName());
  List<SnapshotReportCategory> snapshotReportCategoryList=
apiSanpshotReport.getCategories();
  int i=0;
  for(SnapshotReportCategory snapshotReportCategory:snapshotReportCategoryList){
   System.out.println("Category_"+i);
   System.out.println("categoryName: "+snapshotReportCategory.getCategoryName());
   int j=0;
   ReportNameValuePair[]
reportNameValuePairArr=snapshotReportCategory.getNameValuePairs();
   for(ReportNameValuePair reportNameValuePair:reportNameValuePairArr ){
    System.out.println("ReportNameValuePair_"+j);
    System.out.println("name: "+reportNameValuePair.getName());
    System.out.println("value: "+reportNameValuePair.getValue());
    j++;
   }
   i++;
  }
  }
}
```

Results

The snapshot of the given report context is displayed.

Implementation

Use the userAPIGetInstantDataReport API and pass the contextName, contextValue, and reportId to view the snapshot of the given report context.

See Also

Viewing Available Reports Definition

Viewing Historical Report

Viewing Resource Usage Report

Viewing Tabular Reports

Viewing Tabular Reports

Objective

You can use the userAPIFilterTabularReport API to view a set of tabular reports that are filtered based on a particular context.

Context

To view the tabular reports for a particular context.

Prerequisites

A context must be available in Cisco UCS Director.

REST URL

Request

```
/app/api/rest?formatType=json&opName=userAPIFilterTabularReport&opData=
{param0:"1",param1:"VMware70",param2:"VMS-T0",param3:"VM-ID",param4:"1"}
```

Response

```
{ "serviceResult":{"rows":[{"Cloud":"VMware70","VM_ID":1,"User_Label":"",
"VM_Name":"vm-QA-SR169","Host_Name":null,"IP_Address":"","Image_Id":"vm-QA-SR169",
"Host_Node":"10.28.106.71","Power_Status":"ON","Group_Name":"Default Group",
"vDC":"test vdc","Category":"Discovered VM","Provisioned_Time":"",
"Scheduled_Termination_Time":"","Last_Status_Update":"May 18, 2016 02:35:32 UTC",
"Guest_OS_Type":"Red Hat Enterprise Linux 4 (32-bit)"}],"columnMetaData":null,
"reportParams":{}}, "serviceError":null, "serviceName":"InfraMgr",
"opName":"userAPIFilterTabularReport" }
```

Components

- param0—The name of the context.
- param1—The value of the context.
- param2—The ID of the report.
- param3—Column label.
- param4—Column value.

Code

```
import com.cisco.cuic.api.client.APITabularReport;
import com.cisco.cuic.api.client.CuicServer;
import com.cisco.cuic.api.models.UserAPIGlobal;
import com.cisco.cuic.api.models.UserAPIServiceRequest;

public class TestuserAPIGetTabularReport
{
 public static void main(String[] args)throws Exception
{
 CuicServer api = CuicServer.getAPI("172.22.234.243", "CF87FA987C8F4BBF814F2BB68CA6A823",
 "http", 80);
 UserAPIGlobal instance = new UserAPIGlobal(api);
 APITabularReport report=instance.userAPIFilterTabularReport("0", "All%20Clouds",
"VMS-T0","VM-ID","9");
 System.out.println(report.getRowCount());
 }
}
```

Results

The tabular report for a specific context is filtered and displayed.

Implementation

Use the userAPIFilterTabularReport API and pass the report parameter to view the tabular report.

See Also

Viewing Available Reports Definition

Viewing Historical Report

Viewing Resource Usage Report

Viewing Snapshot Report