# vCloud SDK for .NET Developer's Guide

VMware vCloud SDK 1.0

This document supports the version of each product listed and supports all subsequent versions until the document is replaced by a new edition. To check for more recent editions of this document, see http://www.vmware.com/support/pubs.

EN-000470-00

vmware®

You can find the most up-to-date technical documentation on the VMware Web site at:

http://www.vmware.com/support/

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:

docfeedback@vmware.com



**VMware, Inc.**
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

# Contents

# About This Book

This book, the *vCloud SDK for .NET Developer's Guide*, provides information about using the VMware vCloud® SDK for .NET.

VMware provides several APIs and SDKs for different applications and goals. This book provides information about using the vCloud SDK for .NET for developers who are creating VMware vCloud Director client applications in C# using the .NET framework.

To view the current version of this book as well as all VMware API and SDK documentation, go to http://www.vmware.com/support/pubs/sdk_pubs.html.

## Revision History

This guide is revised with each release of the product or when necessary. A revised version can contain minor or major changes. Table 1 summarizes the significant changes in each version of this guide.

**Table 1.** Revision History

| Revision | Description |
|---|---|
| 01NOV10 | Version 1.0. Amendments and minor corrections to Beta content. |
| 29SEP10 | Version 1.0 (Beta) |

## Intended Audience

This guide is intended for software developers who are building vCloud API applications, including interactive clients of VMware Cloud Director. This guide assumes you are familiar with the C# programming language, the .NET framework, Representational State Transfer (REST) and RESTful programming conventions, the Open Virtualization Format (OVF) Specification, and VMware virtual machine technology. Familiarity with other technologies such as XML, HTTP, and the Windows or Linux operating systems is also assumed.

## VMware Technical Publications Glossary

VMware Technical Publications provides a glossary of terms that might be unfamiliar to you. For definitions of terms as they are used in VMware technical documentation go to http://www.vmware.com/support/pubs.

## Document Feedback

VMware welcomes your suggestions for improving our documentation. Send your feedback to docfeedback@vmware.com.

# Technical Support and Education Resources

The following sections describe the technical support resources available to you. To access the current versions of other VMware books, go to http://www.vmware.com/support/pubs.

## Online and Telephone Support

To use online support to submit technical support requests, view your product and contract information, and register your products, go to http://communities.vmware.com/community/developer.

## Support Offerings

To find out how VMware support offerings can help meet your business needs, go to http://www.vmware.com/support/services.

## VMware Professional Services

VMware Education Services courses offer extensive hands-on labs, case study examples, and course materials designed to be used as on-the-job reference tools. Courses are available onsite, in the classroom, and live online. For onsite pilot programs and implementation best practices, VMware Consulting Services provides offerings to help you assess, plan, build, and manage your virtual environment. To access information about education classes, certification programs, and consulting services, go to http://www.vmware.com/services.

# About the vCloud SDK for .NET 1

The VMware vCloud API provides support for developers who are building interactive clients of VMware vCloud Director using a RESTful application development style. vCloud API clients and servers communicate over HTTP, exchanging representations of vCloud objects. These representations take the form of XML elements. HTTP GET requests are used to retrieve the current representation of an object, HTTP POST and PUT requests are used to create or modify an object, and HTTP DELETE requests are typically used to delete an object.

The vCloud SDK for .NET is a C# language binding for the vCloud API. It provides C# class libraries and a set of example applications. The classes and functions in the library encapsulate the interfaces, objects, and operations that the vCloud API supports, while preserving its RESTful approach and compatibility with the HTTP protocol family.

This *vCloud SDK for .NET Developer's Guide* provides information about setting up the SDK in a development environment, and information about building and running the example applications included in the SDK.

This chapter includes the following topics:



## vCloud Object Taxonomy

The vCloud SDK for .NET defines a set of objects common to cloud computing environments. Figure 1-1 shows the principal object types.

### vCloud Organizations

A vCloud contains one or more organizations. A vCloud organization is a unit of administration for a collection of users, groups, and computing resources. Users authenticate at the organization level, supplying credentials established by an organization administrator when the user was created or imported.

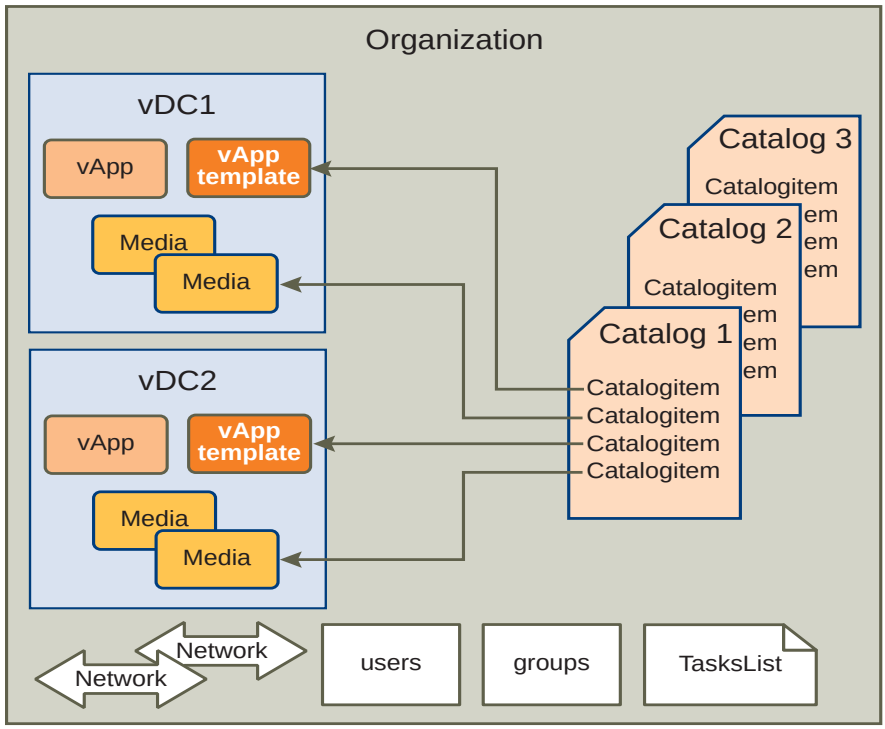
#### vCloud Users and Groups

An organization can contain an arbitrary number of users and groups. Users can be created by the organization administrator or imported from a directory service such as LDAP. Groups must be imported from the directory service. Permissions within an organization are controlled through the assignment of rights and roles to users and groups.

#### vCloud Networks

An organization can be provisioned with one or more networks. These organization networks can be configured to provide services such as DHCP, NAT, and firewalls.

**Figure 1-1.** vCloud Object Taxonomy




## vCloud Virtual Datacenters

A vCloud virtual datacenter (vDC) is an allocation mechanism for resources such as networks, storage, CPU, and memory. In a vDC, computing resources are fully virtualized, and can be allocated based on demand, service level requirements, or a combination of the two.

There are two kinds of vDCs:

- **Provider vDCs**. These vDCs contain all the resources available from the vCloud service provider. Provider vDCs are created and managed by vCloud system administrators.
- **Organization vDCs**. These vDCs provide an environment where virtual systems can be stored, deployed, and operated. They also provide storage for virtual media, such as floppy disks and CD ROMs.

An organization administrator specifies how resources from a provider vDC are distributed to the vDCs in an organization.

## vCloud Catalogs

Catalogs contain references to virtual systems and media images. A catalog can be shared to make it visible to other members of an organization, and can be published to make it visible to other organizations. A vCloud system administrator specifies which organizations can publish catalogs, and an organization administrator controls access to catalogs by organization members.

## vCloud Tasks

Long-running operations initiated by members of an organization create tasks, which are kept on the organization's tasks list.

### Virtual Systems and Media Images in a vCloud

Virtual systems and media images are stored in a vDC and can be included in a catalog. Media images are stored in their native representation (ISO or floppy). Virtual systems are stored as templates, using an open standard format (OVF 1.0). These templates can be retrieved from catalogs and transformed into virtual systems, called vApps, through a process called instantiation, which binds a template's abstract resource requirements to resources available in a vDC. A vApp contains one or more individual virtual machines (`Vm` elements), along with parameters that define operational details such as:

- How the contained virtual machines are connected to each other and to external networks.
- The order in which individual virtual machines are powered on or off.
- End-user license agreement terms for each virtual machine.
- Deployment lease terms (typically inherited from the containing organization) that constrain the vApp's consumption of vDC resources
- Access control information specifying which users and groups can perform operations such as deploy, power on, modify, and suspend on the vApp and the virtual machines it contains.

## vCloud SDK for .NET Design

The vCloud SDK for .NET provides object-specific methods for creating, updating, retrieving, and deleting objects defined by the vCloud API. It also provides methods for operating virtual systems. The SDK includes the following class libraries

- `com.vmware.vcloud.api.rest.schema` provides constructor, getter, and setter methods for all objects defined by the vCloud API.
- `com.vmware.vcloud.sdk` provides methods that create, update, retrieve, and delete vCloud API objects.
- `com.vmware.vcloud.sdk.admin` provides methods that create, update, retrieve, and delete vCloud administrative API objects.
- `com.vmware.vcloud.sdk.admin.extensions` provides methods that create, update, retrieve, and delete vCloud API vSphere extension objects.
- `com.vmware.vcloud.sdk.utility` provides utility methods that simplify the implementation of clients.

Many of the classes implemented in these libraries are wrapper classes whose methods access vCloud API resources using an object reference. Each vCloud API object reference includes the URL (`href` attribute value), resource type, and name properties that define the object. Static methods get resources by passing object references, and act as constructors for SDK wrapper objects.

---

**NOTE** The vCloud SDK for .NET does not provide object lifecycle management. Every wrapper object represents the resource at the time of the GET operation. If a client makes multiple GET requests for the same resource, the client receives multiple representations of the resource wrapped in the helper object. There is no automatic refresh of the client-side representation. It is the client's responsibility to make new requests to get the latest values. To avoid memory leaks, the client must dispose of objects that are not in use.

---

# Setting Up for .NET Development 2

This chapter describes how to prepare for using the vCloud SDK for .NET, how to download the SDK, and how to install and use it.

This chapter includes the following topics:



## Prerequisites

The vCloud SDK for .NET requires the following software to be installed on the development host:

- Microsoft Visual Studio 2008 or later
- Microsoft .NET Framework 3.5 or later
- Additional DLL files, as documented in the README file in the download.

This document and the SDK reference documentation assume that you are familiar with the C# programming language, Microsoft .NET framework and Visual Studio, and have access to an installation of VMware vCloud Director.

In addition, consider the following items:

- Although the vCloud SDK for .NET reference documentation provides information about the vCloud API XML schemas, which define the objects and operations that the SDK supports, familiarity with the details of the underlying objects and operations, as described in the *vCloud API Programming Guide*, can help you understand the structure of vCloud API objects, and how the methods in this SDK operate on those objects.
- Before you can run the examples, you must use the vCloud Director Web console or the vCloud API to create an organization, catalog, and vDC that the examples can use. The organization must have a user account with rights to run the examples. The predefined `CatalogAuthor` role should provide all the necessary rights. For more information about roles and rights, see the *VMware Cloud Director Administrator's Guide.*
- Several of the example programs require you to have an OVF package available on the client host. This package must be uncompressed, and must include exactly one `vmdk` file. For more information about OVF, see the *vCloud API Programming Guide.*

### About SSL Access

In the default configuration, VMware vCloud Director requires vCloud API clients to use SSL. To simplify access to vCloud Director, all SDK examples use a `FakeCertificatePolicy` method that allows the example programs to accept all SSL certificates. Because clients that use this method are inherently insecure, restrict use of this method to example applications running in trusted environments. All of the example applications use the `FakeCertificatePolicy` method.

# Download and Install the vCloud SDK for .NET

You can download the vCloud SDK for .NET from the VMware Web site. The SDK is distributed as a compressed archive named `Vcloud.NetSDK–`*build*`.zip`, where *build* is the build number of the SDK.

### To download and install the vCloud SDK for .NET

1. In a browser, go to http://www.vmware.com/go/vcloudsdkfordotnet.
2. In the **Resources** area of the vCloud SDK for .NET Community page, click the **Download** button.
3. On the **Download** page, log in with your VMware customer credentials.
4. Accept the license agreement to continue.
5. Choose a download option, then click the link to start the download.
6. When the download completes, uncompress the download package into any convenient folder on your computer. Uncompressed, the archive requires about 18 MB of disk space. The package includes the following folders:
   - `Docs`: vCloud SDK for .NET reference documentation in HTML format.
   - `Samples`: Example code demonstrating common use cases associated with programmatically managing virtual infrastructure.
7. Import the package into Visual Studio.
8. See the README file in the download for information about additional DLL files that you must obtain.

# Hello vCloud: A Structured Workflow Example

# 3

This chapter presents an example of using the vCloud SDK for .NET to implement a structured workflow through the lifecycle of a vApp.

This chapter contains the following topics.



## Running the HellovCloud Example

The `HellovCloud` example, included in the `Samples` folder, demonstrates the following operations supported by the vCloud SDK for .NET:

- Logging in to the vCloud
- Uploading an OVF package to create a vApp template
- Adding the vApp template to a catalog
- Instantiating the vApp template to create a vApp
- Operating the vApp

The file `HellovCloud.txt` in the `Samples` folder includes example program input and output.

**NOTE** Before you can run the HellovCloud example, you must build it. For information about building the examples, see "Building the Examples" on page 19.

To run the `HellovCloud` example, use a command of the following form:

```
.Net HellovCloud vCloudApiVersionsURL versionId user@vcloud-organization password orgName vdcName
ovfFileLocation vmdkFileLocation vmdkFileName catalogName
```

The following options are required:

- *vCloudApiVersionsURL* is the base API URL of the vCloud.
- *versionId* is the version of the API to use. Use the value `1.0` with vCloud Director 1.0.
- *username* is the name of a vCloud Director user, in the form *user@vcloud-organization*, who has rights to upload OVF, create vApp templates, create vApps, and operate vApps.

- *password* is the user's password.
- *orgName* is the name of the organization to which the user is authenticating.
- *vdcName* is the name of a vDC in that organization where the user can upload the OVF and deploy the vApp.
- *ovfFileLocation* is the full pathname to the OVF descriptor on the local disk.
- *vmdkFileLocation* is the full pathname to the `vmdk` file referenced in the OVF descriptor.
- *vmdkFileName* is the file name of the `vmdk` file.
- *catalogName* is the name of the catalog in which the vApp template will be catalogued.

For example:

```
.Net HellovCloud https://vcloud.example.com/api/versions 1.0 user@exampleOrg Pa55w0rd exampleOrg
exampleVdc C:\descriptor.ovf C:\disk.vmdk disk.vmdk exampleCatalog
```

## Logging In and Getting an Organization List

Most vCloud API requests must be authenticated by a login request that supplies user credentials in the form required by Basic HTTP authentication (MIME Base64 encoding of a string having the form *user@vcloud-organization:password*). The `vCloudClient` class implements a `login` method that takes the following parameters:

- `userName:` supplied in the form *user@vcloud-organization*
- `password`: the user's password

The `HellovCloud` example uses this method to authenticate to the cloud. The vCloud API returns a list of the organizations to which the user has access, and the `login` method in `HellovCloud` prints this list.

## Getting References to the vDC and Catalog

To instantiate a vApp template and operate the resulting vApp, you need the object references (`href` values) for the catalog in which the vApp template will be listed and the vDC in which the vApp will be deployed. The `Organization` class implements several methods that return references to vDCs and catalogs. `HellovCloud` uses these methods as shown in Example 3-1.

**Example 3-1.** Getting References to the vDC and Catalog

```
public static Vdc FindVdc(string orgName, string vdcName)
{
    try
    {
        ReferenceType orgRef = client.GetOrgRefByName(orgName);
        Organization org = Organization.GetOrganizationByReference(client, orgRef);
        ReferenceType vdcRef = org.GetVdcRefByName(vdcName);
        return Vdc.GetVdcByReference(client, vdcRef);
    }
    ...
}
```

## Upload an OVF Package to Create a vApp Template

The `HellovCloud` command line requires you to supply the name of an OVF descriptor file and the `vmdk` file that it references. This information is used in the `createUploadvAppTemplate` method to upload the OVF descriptor and `vmdk` file, create a vApp template, and return a reference to the template that can be used by other methods in the program.

The `createUploadvAppTemplate` method and the methods it calls from the vCloud SDK for .NET class libraries implement the following workflow to upload the OVF package and create a vApp template.

1 The client POSTs an initial request that specifies a name for the template, a transfer format for the data, and an optional description.

2 The server returns an unresolved (`status="0"`) `vAppTemplate` document that includes an upload URL for the OVF package.

3 The client uses an HTTP PUT request to upload the OVF descriptor to the upload URL.

4 The server reads the descriptor and constructs a `vAppTemplate` object that includes an upload URL for each file listed in the `References` section of the descriptor. While the server is constructing this document, the client makes periodic requests for it and examines the response for additional upload URLs. When the response contains any upload URLs other than the one returned in Step 2, the template is complete.

5 The client uses HTTP PUT requests to upload each of the files.

6 If the OVF package includes a manifest file, the entire upload is validated against the contents of the manifest file.

After all the files are uploaded, and validated if a manifest is present, the server processes the uploads. When processing is complete, the server sets the value of the template's `status` attribute to `8`, indicating that the template is ready for use. This status value indicates that all of the virtual machines in the template are powered off. For more information, see the *vCloud API Programming Guide.*

**Example 3-2.** Upload an OVF Package to Create a vApp Template

```
public static ReferenceType CreateUploadvAppTemplate(Vdc vdc, string ovfFileLocation, string
               vmdkFileLocation, string vmdkFileName)
{
    try
    {
    ...
// create an UploadVappTemplateParams request body and fill in the name and description for the
               vAppTemplate
    UploadVAppTemplateParamsType vappTemplParams = new UploadVAppTemplateParamsType();
    vappTemplParams.Description = "HellovCloudvAppTemplate Description";
    vappTemplParams.name = "HellovCloudvAppTemplate";
    ...
// make the request to the vDC's uploadVappTemplate URL
    VappTemplate vappTemplate = vdc.CreateVappTemplate(vappTemplParams);
    ...
// get the upload:default URL and PUT the descriptor
    FileStream ovfFileInputStream = File.OpenRead(ovfFileLocation);
    vappTemplate.UploadOVFFile(ovfFileInputStream, ovfFileInputStream.Length);
    vappTemplate = VappTemplate.GetVappTemplateByReference(client, vappTemplate.Reference);
    ...
// periodically check the vAppTemplate URL, looking for ovfDescriptorUploaded="true"
    while (!vappTemplate.Resource.ovfDescriptorUploaded)
    {
        System.Threading.Thread.Sleep(5000);
        vappTemplate = VappTemplate.GetVappTemplateByReference(client, vappTemplate.Reference);
    }
    ...
// when the descriptor has been uploaded, upload the VMDK file
    ...
    FileStream vmdkFileInputStream = File.OpenRead(vmdkFileLocation);
    vappTemplate.UploadFile(vmdkFileName, vmdkFileInputStream, vmdkFileInputStream.Length);
    vappTemplate = VappTemplate.GetVappTemplateByReference(client, vappTemplate.Reference);
    while (vappTemplate.Resource.status != 8)
    ...
```

```
// periodically check the vAppTemplate URL. When status="8" upload is complete.
    {
        System.Threading.Thread.Sleep(5000);
        vappTemplate = VappTemplate.GetVappTemplateByReference(client, vappTemplate.Reference);
    }
// return the template URL
    return vappTemplate.Reference;
...
}
```

## Add the vApp Template to a Catalog

After the vAppTemplate is uploaded. the `HellovCloud` example uses its `createNewCatalogItem` method to create a `CatalogItem` object in the catalog whose name was provided on the command line. The `CatalogItem` contains the reference to the template that was returned in Example 3-2.

## Instantiate the vApp Template

When the template is in the catalog, you can instantiate it to create a vApp. The `HellovCloud` example implements a `newvAppFromTemplate` method that has two parameters:

- `vAppTemplateReference`: a reference to the template, which is obtained from the catalog.
- `Vdc`: a reference to the vDC in which to instantiate the template.

With these inputs, `newvAppFromTemplate` constructs a simple `InstantiateVAppTemplateParams` request body, makes the request to the `action/instantiateVAppTemplate` URL of the vDC, and returns a Vapp helper object that contains a reference to the vApp.

**Example 3-3.** Instantiating the vApp Template

```
public static Vapp NewvAppFromTemplate(ReferenceType vAppTemplateReference, Vdc vdc)
{
    try
    {
        ...
// get the href of the OrgNetwork to which we can connect the vApp network, and fail if
// there is no OrgNetwork available
        NetworkConfigurationType networkConfigurationType = new NetworkConfigurationType();
        ...
        if (vdc.GetAvailableNetworkRefs().Count == 0)
        {
            Console.WriteLine("No Networks in vdc to instantiate the vapp");
        }
// specify the NetworkConfiguration for the vApp network
        networkConfigurationType.ParentNetwork = vdc.GetAvailableNetworkRefs().FirstOrDefault();
        networkConfigurationType.FenceMode = FenceModeValuesType.bridged;

        VAppNetworkConfigurationType vAppNetworkConfigurationType = new
                        VAppNetworkConfigurationType();
        vAppNetworkConfigurationType.Configuration = networkConfigurationType;
        if (vdc.GetAvailableNetworkRefs() != null && vdc.GetAvailableNetworkRefs().Count > 0)
        {
            vAppNetworkConfigurationType.networkName =
                            vdc.GetAvailableNetworkRefs().FirstOrDefault().name;
        }
// fill in the NetworkConfigSection
        NetworkConfigSectionType networkConfigSectionType = new NetworkConfigSectionType();
        Msg_Type networkInfo = new Msg_Type();
        networkConfigSectionType.Info = networkInfo;
        List<VAppNetworkConfigurationType> vAppNetworkConfigs = new
                        List<VAppNetworkConfigurationType>();
        vAppNetworkConfigs.Add(vAppNetworkConfigurationType);

        networkConfigSectionType.NetworkConfig = vAppNetworkConfigs.ToArray();
```

```
        InstantiationParamsType instantiationParamsType = new InstantiationParamsType();

// fill in remaining InstantiationParams
        InstantiateVAppTemplateParamsType instVappTemplParamsType = new
                        InstantiateVAppTemplateParamsType();
        instVappTemplParamsType.name = "HellovCloudvApp";
        instVappTemplParamsType.Source = vAppTemplateReference;
        instVappTemplParamsType.InstantiationParams = instantiationParamsType;

// make the request, and get an href to the vApp in return
        Vapp vapp = vdc.InstantiateVappTemplate(instVappTemplParamsType);
        return vapp;
    }
    catch ...
}
```

## Operate the vApp

The `Vapp` class includes methods that perform operations on the vApp. The majority of these operations return a `Task` object that tracks the progress of the operation. The `HellovCloud` example uses these methods to cycle the vApp through the following states:

1. Deploy the vApp using the `vapp.Deploy()` method.
2. Power on the vApp using the `vapp.PowerOn()` method.
3. Suspend the vApp using the `vapp.Suspend()` method.
4. Power off the vApp using the `vapp.PowerOff()` method.
5. Undeploy the vApp using the `vapp.Undeploy()` method.
6. Delete the vApp using the `vapp.Delete()` method.

# Using the Examples 4

The vCloud SDK for .NET includes example programs that demonstrate how to use the SDK to develop client applications. The examples are in the `Samples` folder of the SDK download.

This chapter contains the following topics:



## Building the Examples

**To build the examples in Visual Studio**

1 Double-click the file `samples.sln`.

2 Click **Build > Build Solution**.

## Running the Examples

Examples listed in Table 4-1 can be run by any user with rights to create and modify catalog items and vApps.

**Table 4-1.** User API Examples

| Example Name | Description |
|---|---|
| CatalogInventory | Lists name and href for all items in all catalogs in the organization. |
| CatalogItemCRUD | Create, retrieve, update, or delete a catalog item. |
| DiskCRUD | Create, retrieve, update, or delete a virtual hard disk in a `Vm` object. |
| ListAllvApps | List all vApps in a vDC by name and `href`. |
| ThreadSample | Examples of how to implement multi-threaded client applications that execute multiple requests in parallel. |
| VdcInventoryCRUD | List name and `href` for all vApps, vApp templates, and media images in all vDCs in the organization. |

Examples listed in Table 4-2 require organization administrator privileges.

**Table 4-2.** Administrative API Examples

| Example Name | Description |
|---|---|
| CatalogCRUD | Create, retrieve, update, or delete a catalog. |
| OrganizationCRUD | Create, retrieve, update, or delete an organization. Requires system administrator privileges. |
| OrgNetworkCRUD | Create, retrieve, update, or delete an organization network. |
| RoleCRUD | Create, retrieve, update, or delete a role. |
| UserCRUD | Create, retrieve, update, or delete a local user. |
| VdcCRUD | Create, retrieve, update, or delete a vDC. |

Each of the example folders includes a `.txt` file that provides an example of example program input and output.

# Index