vCloud SDK for Java Developer's Guide

vCloud SDK for Java 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-000362-00

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

VMware provides several different APIs and SDKs for various applications and goals. This book provides information about using the vCloud SDK for Java for developers that are interested in creating client applications for managing VMware vCloud components available on VMware ESX, VMware ESXi, and VMware vCenter Server systems.

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
30AUG10	Version 1.0

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 Java programming language, Representational State Transfer (REST) and RESTful programming conventions, the Open Virtualization Format Specification, and VMware Virtual machine technology. Familiarity with other widely-deployed 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 Java 1

The VMware vCloud API provides support for developers who are building interactive clients of VMware Cloud 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 Java is a Java language binding for the vCloud API. It uses the JAXB framework to create Java classes for the resources defined in the vCloud API XML schemas. The vCloud SDK for Java provides classes and methods that encapsulate the interfaces, objects, and operations supported by the vCloud API while preserving its RESTful programming style and compatibility with the HTTP protocol family.

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

This chapter includes these topics:

vCloud SDK for Java Design

The vCloud SDK for Java provides object-specific methods for creating, updating, retrieving, and deleting objects defined by the vCloud API. It also includes wrapper classes that provide:

- Helpers for REST communication and Java object representation of the XML resources defined by the vCloud API.
- Helper methods that can assist in client development

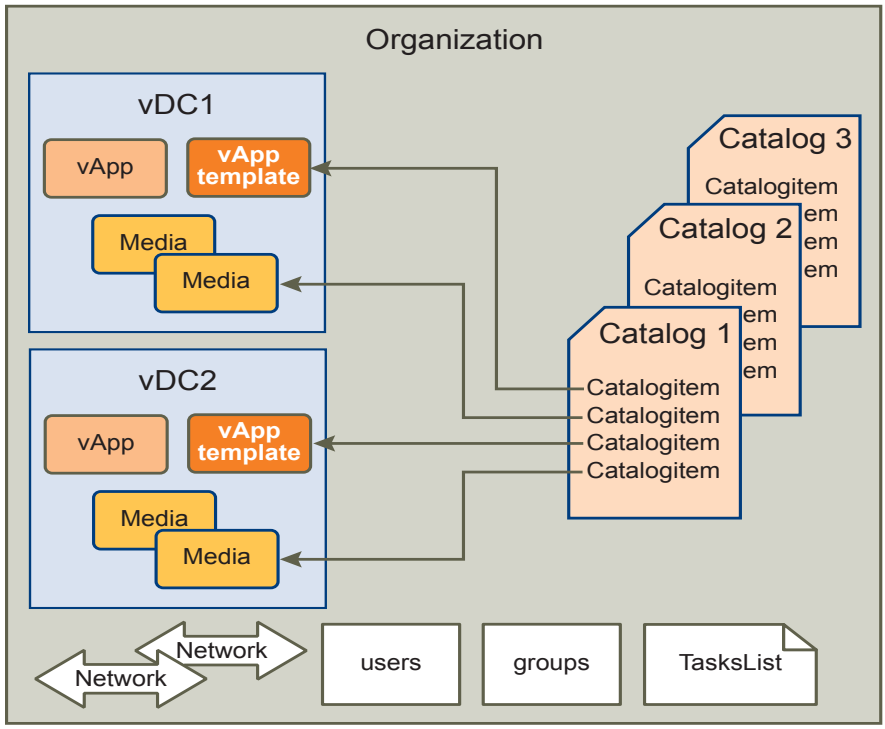
Every wrapper class has methods that access vCloud API resources using a reference. A reference object contains the href, resource type, and name properties. Static methods get resources directly by passing the reference, and act as constructors or factories for SDK wrapper objects.

NOTE The vCloud SDK for Java 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.

vCloud Object Taxonomy

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

Figure 1-1. vCloud Object Taxonomy



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.

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.

Setting Up for Java Development 2

This chapter includes these topics:

Prerequisites

The vCloud SDK for Java requires JDK 6 or later (the SDK and samples were developed using JDK 1.6.0_14-b08). This document and the SDK reference documentation assume that you are familiar with the Java programming language and have access to an installation of VMware Cloud Director.

In addition, you should consider the following:

- Although the vCloud SDK for Java javadoc 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 samples, you must use the Cloud Director web console or the vCloud API to create an organization, catalog, and vDC that the samples can use. The organization must have a user account with rights to run the samples. The predefined role `CatalogAuthor` should provide all the necessary rights. For more information about roles and rights, see the *VMware Cloud Director Administrator's Guide.*
- Several of the sample programs, including `HellovCloud.java`, require you to have an OVF package available on the client host. This package must be uncompressed, and must specify a single `vmdk` file. For more information about OVF, see the *vCloud API Programming Guide.*

Download the vCloud SDK for Java Package

The vCloud SDK for Java is distributed as a zipped archive. Unzipped, it requires about 14.1 MB of disk space. The package includes the following files:

- The SDK package includes precompiled client-side libraries (`vcloud-java-sdk-1.0.jar`, `rest-api-schemas-1.0.0.jar`). The `vcloud-java-sdk-1.0.jar` file contains the vCloud Java SDK classes and methods. The `rest-api-schemas-1.0.0.jar` file contains JAXB-based Java classes for the resources defined in the vCloud API.
- Sample code (`vcloud-java-sdk-samples-1.0-sources.jar`) demonstrating common use cases associated with programmatically managing virtual infrastructure. The samples include Java source code, compiled Java class files, and text examples of program inputs and outputs.

- SDK reference documentation in javadoc form (in the `apidocs` folder) that provides object type definitions, properties, method signatures, and similar information about the vCloud SDK for Java.
- Access to technical publications, including the *vCloud SDK for Java Developer's Guide* (this book), which helps you setup your development environment and run sample applications using Java.

Import the SDK Into Your Java IDE

You can import the vCloud SDK for Java into a Java IDE.

1 Unzip the package (use `gzip` or a similar program).

2 In the unzipped package, open the `vcloud-java-sdk-all` folder and unzip the samples jar `vcloud-java-sdk-samples-1.0-sources.jar`.

3 Import the contents of the `vcloud-java-sdk-all` folder into your Java IDE.

About SSL Access

In the default configuration, VMware Cloud Director requires vCloud API clients to use SSL. To simplify access to Cloud Director, all SDK samples use a `FakeSSLSocketFactory` class that allows the sample programs to accept all SSL certificates. Because clients that use the `FakeSSLSocketFactory` class are inherently insecure, you should restrict use of this method to sample applications, and only in trusted environments. All of the sample applications use the `FakeSSLSocketFactory` class.

Client applications built with this SDK can enable the use of SSL certificates by either importing certificates into a keystore or implementing a custom socket factory that accepts certificates from the server. Client applications should not use the `FakeSSLSocketFactory` class.

Hello vCloud: A Structured Java Workflow Example

3

This chapter presents an example of using the vCloud SDK for Java to implement a structured workflow through the lifecycle of a vApp. It contains the following topics.

Running the HellovCloud Sample

The `HellovCloud.java` sample, included in the `samples` folder of `vcloud-java-sdk-samples-1.0-sources.jar`, demonstrates a number of the operations supported by the vCloud SDK for Java:

- 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 that folder includes sample input and output.

The examples shown in this section are extracted from the `HellovCloud.java` sample.

To run the `HellovCloud.java` sample, use the following command line.

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

where:

- *vCloudApiVersionsURL* is the base API URL of the vCloud.
- *versionId* is the version of the API to use (always 1.0 for this release).
- *username* is the name of a Cloud 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:

```
java HellovCloud https://vcloud/api/versions 1.0 user@SampleOrg Pa55w0rd SampleOrg SampleVdc
C:\descriptor.ovf C:\disk.vmdk disk.vmdk SampleCatalog
```

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 two parameters:

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

As shown in the excerpt in Example 3-1, `HellovCloud.java` 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.java` prints this list. In the typical case, this list has a single member, the organization that was supplied in the `userName` parameter.

Example 3-1. Logging In and Getting an Organization List

```
public static void login(String vCloudVersionsURL, String versionId,String username, String
    password) ...
    {
        ...
        System.out.println("Organizations:");
        System.out.println("--------------");
        for (String organizationName : organizationsMap.keySet())
        System.out.println("" + organizationName);
    } else {
        System.out.println(Try Logging in with valid details");
        System.exit(0);
    }
```

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 entered and the vDC in which the vApp will be deployed. The `Organization` class implements several methods that return references to vDCs and catalogs. `HellovCloud.java` uses these methods as shown in Example 3-2.

Example 3-2. Getting References to the vDC and Catalog

```
public static Vdc findVdc(String orgName, String vdcName) throws VCloudException {
    ReferenceType orgRef = vcloudClient.getOrgRefByName(orgName);
    Organization org = Organization.getOrganizationByReference(vcloudClient, orgRef);
    ReferenceType vdcRef = org.getVdcRefByName(vdcName);
    return Vdc.getVdcByReference(vcloudClient, vdcRef);
}
...
```

```
public static ReferenceType findCatalogRef(String orgName,String catalogName) throws
    VCloudException {
    ReferenceType orgRef = vcloudClient.getOrgRefByName(orgName);
    Organization org = Organization.getOrganizationByReference(vcloudClient, orgRef);
    return org.getCatalogRefByName(catalogName);
}
```

Upload an OVF Package to Create a vApp Template

The `HellovCloud.java` 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 Java 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 package descriptor (the .ovf file) 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 beyond the one returned in Step 2, 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 3-3. Upload an OVF Package to Create a vApp Template

```
public static ReferenceType createUploadvAppTemplate(Vdc vdc, String ovfFileLocation, String
    vmdkFileLocation, String vmdkfileName) throws VCloudException, FileNotFoundException,
    InterruptedException {
    ...
//fill in the upload params with name and Description
    UploadVAppTemplateParamsType vappTemplParams = new UploadVAppTemplateParamsType();
    vappTemplParams.setDescription("HellovCloudvAppTemplate Description");
    vappTemplParams.setName("HellovCloudvAppTemplate");
    ...
//get the action/uploadVappTemplate link
    VappTemplate vappTemplate = vdc.createVappTemplate(vappTemplParams);
    ...
//upload the OVF descriptor
    vappTemplate.uploadOVFFile(ovfFileInputStream, ovfFile.length());
    ...
//using the href of the new vAppTemplate, check to see if the descriptor
// upload is complete (ovfDescriptorUploaded=true)
    vappTemplate = VappTemplate.getVappTemplateByReference(vcloudClient,
        vappTemplate.getReference());
    while (!vappTemplate.getResource().isOvfDescriptorUploaded()) {
        Thread.sleep(5000);
```

```
    ...
//get the upload:default URI for the vmdk file and PUT the serialized bits
      vappTemplate.uploadFile(vmdkfileName, vmdkFileInputStream, vmdkFile .length());
    ...
//check the status. when status=8, template is complete
    while (vappTemplate.getResource().getStatus() != 8) {
        Thread.sleep(5000);
    vappTemplate = VappTemplate.getVappTemplateByReference(vcloudClient,
        vappTemplate.getReference());
      }
    ...
//return the href of the vAppTemplate
    return vappTemplate.getReference();
}
```

Add the vApp Template to a Catalog

After the vAppTemplate has been uploaded, `HellovCloud.java` 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 3-3.

Instantiate the vApp Template

Now that we have the template in the catalog, we can instantiate it to create a vApp. `HellovCloud.java` implements a `newvAppFromTemplate` method that has two parameters:

- `vAppTemplateReference`: a reference to the template (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 (among other things) a reference to the vApp.

Example 3-4. Instantiating the vApp Template

```
public static Vapp newvAppFromTemplate(ReferenceType vAppTemplateReference,Vdc vdc) throws
    VCloudException {
    ...
//get the href of the OrgNetwork to which we can connect the vApp network
    NetworkConfigurationType networkConfigurationType = new NetworkConfigurationType();
    if (vdc.getAvailableNetworkRefs().size() == 0) {
            System.out.println("No Networks in vdc to instantiate the vapp");
            System.exit(0);
    }
//specify the NetworkConfiguration for the vApp network
    networkConfigurationType.setParentNetwork(vdc.getAvailableNetworkRefs()
        .iterator().next());
    networkConfigurationType.setFenceMode(FenceModeValuesType.BRIDGED);

    VAppNetworkConfigurationType vAppNetworkConfigurationType = new
        VAppNetworkConfigurationType();
    vAppNetworkConfigurationType.setConfiguration(networkConfigurationType);
    vAppNetworkConfigurationType.setNetworkName("new network");
//fill in the NetworkConfigSection
    NetworkConfigSectionType networkConfigSectionType = new NetworkConfigSectionType();
    MsgType networkInfo = new MsgType();
    networkInfo.setMsgid("1");
    networkInfo.setValue("network info");
    networkConfigSectionType.setInfo(networkInfo);
    List<VAppNetworkConfigurationType> vAppNetworkConfigs = networkConfigSectionType
        .getNetworkConfig();
    vAppNetworkConfigs.add(vAppNetworkConfigurationType);
//fill in remaining InstatiationParams
    InstantiationParamsType instantiationParamsType = new InstantiationParamsType();
```

```
    List<JAXBElement<? extends SectionType>> sections = instantiationParamsType
        .getSection();
    sections.add(new ObjectFactory()
        .createNetworkConfigSection(networkConfigSectionType));
//create the request body (InstantiateVAppTemplateParams)
    InstantiateVAppTemplateParamsType instVappTemplParamsType = new
        InstantiateVAppTemplateParamsType();
    instVappTemplParamsType.setName("HellovCloudvAppp");
    instVappTemplParamsType.setSource(vAppTemplateReference);
    instVappTemplParamsType.setInstantiationParams(instantiationParamsType);
//make the request, and get an href to the vApp in return
    Vapp vapp = vdc.instantiateVappTemplate(instVappTemplParamsType);
    return vapp;
```

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. `HellovCloud.java` uses a number of these methods to cycle the vApp through the following states:

1 deploy: `deploy()`

2 power on: `powerOn()`

3 suspend: `suspend()`

4 power off: `powerOff()`

5 undeploy: `undeploy()`

6 delete: `delete()`

Overview of Packages and Samples

4

The vCloud SDK for Java includes sample application code and complete reference documentation on all packages, classes, and methods in the SDK. This chapter is an introductory tour of those items.

Packages

The SDK includes four packages that support the JAXB bindings to the vCloud API XML schemas and three packages that implement create, retrieve, update, and delete (CRUD) operations on the objects represented by those schemas.

Table 4-1. Packages in the vCloud SDK for Java

Package Name	Description
com.vmware.vcloud.sdk	Defines the classes that implement the user API. Objects accessed by these classes are typically readable by all users, and can be modified by users with appropriate rights.
com.vmware.vcloud.sdk.admin	Defines the classes that implement the administrative API. Objects accessed by these classes are typically readable by all users, but can be created and modified only by a system administrator, organization administrator or other privileged user. Many of these objects extend types that are defined in the user API.
com.vmware.vcloud.sdk.admin.extension	Defines the classes that implement extensions that support operations on the vSphere Platform. All vSphere platform operations are restricted to the system administrator.
com.vmware.vcloud.api.rest.schema, com.vmware.vcloud.api.rest.schema.extension com.vmware.vcloud.api.rest.schema.ovf com.vmware.vcloud.api.rest.schema.versioning org.w3._2001.xmlschema	XML schemas used by JAXB binding

Samples

In addition to `HellovCloud.java` (see "Hello vCloud: A Structured Java Workflow Example" on page 13), the SDK samples directory (`vcloud-java-sdk-samples-1.0-sources\com\vmware\vcloud\sdk\samples`) includes several samples that demonstrate how you can use the vCloud SDK for Java to develop client applications. Samples listed Table 4-2 can be run by any user with rights to create and modify catalog items and vApps. Samples listed Table 4-3 require organization administrator privileges.

Table 4-2. User API Samples

Sample Name	Description
CatalogInventorySample.java	Lists name and href for all items in all catalogs in the organization.
CatalogItemCRUD.java	Create, retrieve, update, or delete a catalog item.
DiskCRUD.java	Create, retrieve, update, or delete a virtual hard disk in a `Vm` object.

Table 4-2. User API Samples (Continued)

Sample Name	Description
ListAllvApps.java	List all vApps in a vDC by name and href
ThreadSample.java	Examples of how to implement multi-threaded client applications that execute multiple requests in parallel.
VdcInventorySample.java	List name and `href` for all vApps, vApp templates, and media images in all vDCs in the organization

Table 4-3. Administrative API Samples

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

Index