# vCloud SDK for PHP Developer's Guide

vCloud Director 5.1

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-000824-00

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

# vCloud SDK for PHP Developer's Guide

The *vCloud SDK for PHP Developer's Guide* provides information about the PHP SDK for version 5.1 of the vCloud API.

VMware provides APIs and SDKs for various applications and goals. This guide provides information about the vCloud API for developers who are interested in creating RESTful clients of VMware vCloud Director.

## Revision History

The *vCloud SDK for PHP Developer's Guide* is revised with each release of the product or when necessary. A revised version can contain minor or major changes.

**Table 1.** Revision History

| Revision Date | Description |
|---|---|
| 10SEP12 | API Version 5.1 |
| 01SEP11 | API Version 1.5 |
| 30AUG10 | API Version 1.0 |

## Intended Audience

This guide is intended for software developers who are building VMware Ready Cloud Services, including interactive clients of VMware vCloud Director. You should be familiar with the PHP programming language, representational State Transfer (REST) and RESTful programming conventions, the Open Virtualization Format Specification, and VMware Virtual machine technology. You should also be familiar with other widely deployed technologies such as XML, HTTP, and the Windows or Linux operating system.

# About the VMware vCloud API

1

The VMware vCloud API provides support for developers who are building interactive clients of VMware vCloud Director using a RESTful application development style.

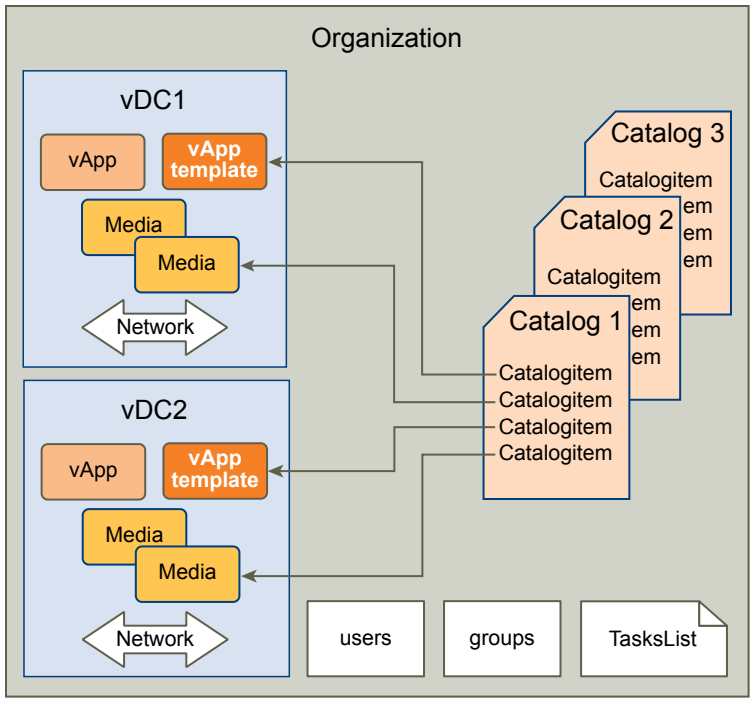
vCloud API clients and vCloud Director servers communicate over HTTP, exchanging representations of vCloud objects. These representations take the form of XML elements. You use HTTP GET requests to retrieve the current representation of an object, HTTP POST and PUT requests to create or modify an object, and HTTP DELETE requests to delete an object.

This chapter includes the following topics:

# Object Taxonomy

The vCloud API defines a set of objects common to cloud computing environments. An understanding of these objects, their properties, and their relationships is essential to using the vCloud API.

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




vCloud API objects have the following high-level properties:

**Organizations**

A cloud can contain one or more organizations. Each organization is a unit of administration for a collection of users, groups, and computing resources. Users authenticate at the organization level, supplying credentials established when the user was created or imported. User credentials are authenticated by the organization's identity provider, which can be either the integrated identity provider included in vCloud Director or an external SAML-based identity provider.

**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 an LDAP directory service or SAML-based identity provider. Groups must be imported. Permissions within an organization are controlled through the assignment of rights and roles to users and groups.

**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 administrators in other organizations. A system administrator specifies which organizations can publish catalogs, and an organization administrator controls access to catalogs by organization members.

### Organization vDCs

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

### Organization vDC Networks

An organization vDC can be provisioned with one or more networks. These organization vDC networks can be configured to provide direct or routed connections to external networks, or can be isolated from external networks and other organization vDC networks. Routed connections require an Edge Gateway and network pool in the vDC. The Edge Gateway provides firewall, network address translation, static routing, VPN, and load balancing services.

### Virtual Systems and Media Images

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 initially 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, including:

- 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 consumption of vDC resources by the vApp.
- 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 that it contains.

### Tasks

Asynchronous operations that members of an organization initiate are tracked by task objects, which are kept on the organization's tasks list.

# Objects, References, and Representations

The vCloud API represents objects as XML documents in which object properties are encoded as elements and attributes with typed values and an explicit object hierarchy defined by an XML schema.

XML representations of first-class vCloud API objects, such as the objects in Figure 1-1, include these attributes.

**id** The object identifier, expressed in URN format. The value of the `id` attribute uniquely identifies the object, persists for the life of the object, and is never reused. The `id` attribute value is intended to provide a context-free identifier that can be used with the vCloud API `entityResolver` and is also suitable for use by clients that need to access the object using a different API.

**type** The object type, specified as a MIME content type.

**href** An object reference, expressed in URL format. Because this URL includes the object identifier portion of the `id` attribute value, it uniquely identifies the object, persists for the life of the object, and is never reused. The value of the `href` attribute is a reference to a view of the object, and can be used to access a representation of the object that is valid in a particular context. Although URLs have a well-known syntax and a well-understood interpretation, a client should treat each `href` as an opaque string. The rules that govern how the server constructs `href` strings might change in future releases.

## Example: Object id, type, and href Attributes

This XML fragment, extracted from the representation of a vApp, shows its `id`, `type`, and `href` attributes.

```
<VApp
   ...
   id="urn:vcloud:vapp:490af534-1491-452e-8ed6-a5eb54447dac"
   type="application/vnd.vmware.vcloud.vApp+xml"
   href="https://vcloud.example.com/api/vApp/vapp-490af534-1491-452e-8ed6-a5eb54447dac"
   ... >
   ...
</VApp>
```

# Links and Link Relations

The vCloud API makes extensive use of `Link` elements to provide references to objects and the actions that they support. These elements are the primary mechanism by which a server tells a client how to access and operate on an object.

The server creates `Link` elements in a response body. They are read-only at the client. If a request body includes a `Link` element, the server ignores it.

## Attributes of a Link Element

In the XML representation of a vCloud object, each `Link` element has the following form:

```
<Link rel="relationship"
   type="application/vnd.vmware.vcloud.object_type+xml"
   href="URL"
   name="string"/>
```

Attribute values in a `Link` element supply the following information:

**rel** Defines the relationship of the link to the object that contains it. A relationship can be the name of an operation on the object, a reference to a contained or containing object, or a reference to an alternate representation of the object. The relationship value implies the HTTP verb to use when you use the link's `href` value as a request URL.

**type** The object type, specified as a MIME content type, of the object that the link references. This attribute is present only for links to objects. It is not present for links to actions.

**href** An object reference, expressed in URL format. Because this URL includes the object identifier portion of the `id` attribute value, it uniquely identifies the object, persists for the life of the object, and is never reused. The value of the `href` attribute is a reference to a view of the object, and can be used to access a representation of the object that is valid in a particular context. Although URLs have a well-known syntax and a well-understood interpretation, a client should treat each `href` as an opaque string. The rules that govern how the server constructs `href` strings might change in future releases.

**name** The name of the referenced object, taken from the value of that object's `name` attribute. Action links do not include a `name` attribute.

**Table 1-1.** Link Relationships and HTTP Request Types

| rel Attribute Value | Action or Relationship Description | Implied HTTP Verb |
|---|---|---|
| abort | Abort this blocking task. | POST |
| add | Add an item to this container. | POST |
| alternate | References an alternate representation of this object. | GET |
| answer | Provide user input requested by a virtual machine. | POST |
| authorization:check | Check whether an extension service operation is authorized for an entity. | POST |
| blockingTask | A list of pending blocking task requests in this cloud. | GET |
| bundle:upload | Upload an extension service localization bundle. | PUT |
| bundles:cleanup | Remove unused extension service localization bundles. | POST |
| catalogItem | References the CatalogItem object that refers to this object. | GET |
| certificate:reset | Removes the SSL certificate used by this service. | POST |
| certificate:update | Updates the SSL certificate used by this service. | POST |
| checkCompliance | Check that this virtual machine is using a storage profile of the intended type. | POST |
| consolidate | Consolidate this virtual machine. | POST |
| controlAccess | Apply access controls to this object. | POST |
| copy | Reserved | N/A |
| deploy | Deploy this vApp. | POST |
| disable | Disable this object. | POST |
| discardState | Discard the suspended state of this virtual machine. | POST |
| disk:attach | Attach an independent disk to this virtual machine. | POST |
| disk:detach | Detach an independent disk from this virtual machine. | POST |
| down | References an object contained by this object. | GET |

**Table 1-1.** Link Relationships and HTTP Request Types (Continued)

| rel Attribute Value | Action or Relationship Description | Implied HTTP Verb |
|---|---|---|
| down:aclRules | Retrieve the ACL rules for this resource class action. | GET |
| down:apidefinitions | Retrieve the API definitions for this extension service. | GET |
| down:apiDefinitions | Retrieve the API definitions for this extension service. | GET |
| down:apiFilters | Retrieve the API filters for this extension service. | GET |
| down:extensibility | Add an extension service to the system. | POST |
| down:fileDescriptors | Retrieve file descriptors for extension services APIs | GET |
| down:files | Retrieve files for extension services APIs | GET |
| down:resourceClassActions | Retrieve the actions defined for this extension service resource class. | GET |
| down:resourceClasses | Retrieve the resource classes defined by this extension service. | GET |
| down:service | | |
| down:serviceLinks | Retrieve the service links defined by this extension service. | GET |
| down:serviceResources | Retrieve the list of extension service resources of this class. | |
| down:services | Retrieve the list of registered extension services. | GET |
| download:alternate | Reserved | N/A |
| download:default | References the default location from which this file can be downloaded. | GET |
| download:identity | References the extended OVF descriptor of this vApp template. The extended OVF descriptor contains additional information such as MAC address, BIOS UUID, and NetworkConfigSection | GET |
| edgeGateway:configureServices | Update the network services offered by this Edge Gateway. | PUT |
| edgeGateway:reapplyServices | Reapply (after an update) the network services offered by this Edge Gateway. | POST |
| edgeGateway:redeploy | Redeploy the vShield Edge supporting this Edge Gateway. | POST |
| edgeGateway:syncSyslogSettings | Synchronize syslog server addresses used by this Edge Gateway with system defaults. | POST |
| edgeGateway:upgrade | Upgrade the backing configuration of this Edge Gateway from compact to full. | POST |
| edgeGateways | List the Edge Gateway objects in this organization vDC. | GET |
| edit | Modify this object, typically by replacing its current representation with the one in the request body. | PUT |
| enable | Enable this object. | POST |
| enterMaintenanceMode | Put this virtual machine into maintenance mode. | POST |
| entity | Retrieve a representation of the object on which an operation triggered this notification. | GET |
| entityResolver | Retrieve an object id as a context-free Entity element. | GET |
| event:create | Create an event in an this organization's event stream. | POST |
| exitMaintenanceMode | Take this virtual machine out of maintenance mode. | POST |
| fail | Fail this blocking task. | POST |
| firstPage | Reference to the first page of a paginated response. | GET |

**Table 1-1.** Link Relationships and HTTP Request Types (Continued)

| rel Attribute Value | Action or Relationship Description | Implied HTTP Verb |
|---|---|---|
| installVmwareTools | Install VMware Tools on this virtual machine. | POST |
| keystore:reset | Removes the keystore used by this service. | POST |
| keystore:update | Updates the keystore used by this service. | POST |
| keytab:reset | Removes the keytab used by this service. | POST |
| keytab:update | Updates the keytab used by this service. | POST |
| lastPage | Reference to the last page of a paginated response. | GET |
| media:ejectMedia | Eject virtual media from a virtual device. | POST |
| media:insertMedia | Insert virtual media into a virtual device. | POST |
| merge | Merge one or more Provider vDCs with this Provider vDC. | POST |
| migrateVms | Migrate virtual machines from this resource pool to a different one. | POST |
| move | Reserved | N/A |
| nextPage | Reference to the next page of a paginated response. | GET |
| orgVdcNetworks | List the organization vDC networks supported by this Edge Gateway. | GET |
| ova | Reserved | N/A |
| ovf | References the OVF descriptor of this vApp template. | GET |
| power:powerOff | Power off this vApp or virtual machine. | POST |
| power:powerOn | Power on this vApp or virtual machine. | POST |
| power:reboot | Reboot this vApp or virtual machine. | POST |
| power:reset | Reset this vApp or virtual machine. | POST |
| power:shutdown | Shut down this vApp or virtual machine. | POST |
| power:suspend | Suspend this vApp or virtual machine. | POST |
| previousPage | Reference to the previous page of a paginated response. | GET |
| publish | Publish this catalog. | POST |
| recompose | Recompose this vApp to add, remove, or reconfigure virtual machines. | POST |
| reconfigureVm | Update multiple sections of a virtual machine. | POST |
| reconnect | Reconnect this vCenter Server to the system. | POST |
| refreshStorageProfiles | Refresh the list of storage profiles that exist on the vCenter service backing this Provider vDC. | POST |
| refreshVirtualCenter | Refresh the representation of this vCenter server | POST |
| register | Register a VCenter Server with the system. | POST |
| relocate | Relocate this virtual machine. | POST |
| remove | Remove this object. | DELETE |
| remove:force | Force removal of this object. | DELETE |
| repair | Repair this host or network. | POST |
| resourcePoolVmList | List the virtual machines using this resource pool. | GET |
| resume | Resume this blocking task. | POST |
| rights | List the service-specific rights created by this extension service. | GET |

**Table 1-1.** Link Relationships and HTTP Request Types (Continued)

| rel Attribute Value | Action or Relationship Description | Implied HTTP Verb |
|---|---|---|
| rights:cleanup | Remove service-specific rights no longer used by any extension service. | POST |
| screen:acquireTicket | Retrieve a screen ticket for this virtual machine. | GET |
| screen:thumbnail | Retrieve a thumbnail view of the screen of this virtual machine. | GET |
| shadowVms | List shadow virtual machines associated with the virtual machines in this vApp template. | GET |
| snapshot:create | Create a snapshot of the virtual machines in this vApp. | POST |
| snapshot:removeAll | Remove all snapshots created for the virtual machines in this vApp. | POST |
| snapshot:revertToCurrent | Revert all virtual machines in this vApp to their current snapshot. | POST |
| storageProfile | References the storage profile for this object. | GET |
| syncSyslogSettings | Synchronize syslog server addresses used by this vApp network with system defaults. | POST |
| task | Retrieve the blocking task that triggered this notification. | GET |
| task:cancel | Cancel this task. | POST |
| task:create | Create a task object. | POST |
| task:owner | Reference to the owner of a task. | GET |
| truststore:reset | Remove the truststore used by this service. | POST |
| truststore:update | Update the truststore used by this service. | PUT |
| undeploy | Undeploy this vApp. | POST |
| unlock | Unlock this user account. | POST |
| unregister | Unregister this vCenter Server. | POST |
| up | References an object that contains this object. | GET |
| update:resourcePools | Update the resource pools of this Provider vDC | POST |
| updateProgress | Request an update of this task's progress. | POST |
| upgrade | Upgrade this ESX/ESXi host. | POST |
| upload:alternate | Reserved | N/A |
| upload:default | References the default location to which this object can be uploaded. | PUT |
| vSphereWebClientUrl | A URL that you can use to view this object with the vSphere Web Client | GET |

# Client Workflow Overview

vCloud API clients implement a RESTful workflow, making HTTP requests to the server and retrieving the information they need from the server's responses.

## About RESTful Workflows

REST, an acronym for Representational State Transfer, describes an architectural style characteristic of programs that rely on the inherent properties of hypermedia to create and modify the state of an object whose serialized representation is accessible at a URL.

If a URL of such an object is known to a client, the client can use an HTTP GET request to retrieve the representation of the object. In the vCloud API, this representation is an XML document. In a RESTful workflow, representations of object state are passed back and forth between a client and a service with the explicit assumption that neither party need know anything about an object other than what is presented in a single request or response. The URLs at which these documents are available often persist beyond the lifetime of the request or response that includes them. The other content of the documents is nominally valid until the expiration date noted in the HTTP `Expires` header.

## vCloud REST API Workflows

Application programs written to a REST API use HTTP requests that are often executed by a script or other higher-level language to make remote procedure calls that create, retrieve, update, or delete objects that the API defines. In the vCloud REST API, these objects are defined by a collection of XML schemas. The operations themselves are HTTP requests, and so are generic to all HTTP clients.

To write a RESTful client, you must understand only the HTTP protocol and the semantics of XML, the transfer format that the vCloud API uses. To use the vCloud API effectively in such a client, you need to know only a few things:

- The set of objects that the API supports, and what they represent; for example, what is a vDC and how does it relate to an organization or catalog?
- How the API represents these objects; for example, what does the XML schema for an `Org` look like? What do the individual elements and attributes represent?
- How a client refers to an object on which it wants to operate; for example, where are the links to objects in a vDC? How does a client obtain and use them?

You can find this information in the vCloud API XML schemas. The XML elements, attributes, and composition rules defined in these schemas represent the data structures of objects in the cloud. A client can read an object by making an HTTP GET request to the object's URL. A client can create or modify an object with an HTTP PUT or POST request that includes a new or changed XML body document for the object. A client can usually delete an object with an HTTP DELETE request.

The vCloud API schema reference includes detailed information about the XML representations of all vCloud API objects and examples of HTTP requests that operate on those objects. See "About the Schema Reference," on page 17.

## RESTful Workflow Patterns

All RESTful workflows follow a common pattern.

1. Make an HTTP request, typically GET, PUT, POST, or DELETE. The target of this request is either a well-known URL such as the vCloud API `versions` URL, or a URL obtained from the response to a previous request. For example, a GET request to an organization URL returns links to catalog and vDC objects that the organization contains.
2. Examine the response, which always includes an HTTP response code and usually includes a body. In the vCloud API, a response body is an XML representation of an object, including elements and attributes that represent object properties, links that implement operations on the object or provide references to contained or containing objects and, if the object is being created or modified, an embedded `Task` object that tracks the progress of the creation or modification. The response also includes an HTTP response code, which indicates whether the request succeeded or failed, and might be accompanied by a URL that points to a location from which you can retrieve additional information.

These operations can repeat, in this order, for as long as necessary.

## vCloud API REST Requests

To retrieve object representations, clients make HTTP requests to object references. The server supplies these references as `href` attribute values in responses to GET requests.

Every cloud has a well-known URL from which an unauthenticated user can retrieve a `SupportedVersions` document, which lists each version of the of vCloud API that the server supports. For each version, the response lists the names and MIME types of the complex types defined in the version's XML namespace, and the version login URL. A system administrator can use that URL to authenticate to the cloud by logging in to the `System` organization. An authenticated user can discover other vCloud API URLs by making GET requests to URLs retrieved from the login response, and the URLs contained in responses to those requests.

Requests are typically categorized in terms of the type of requested operation: create, retrieve, update, and delete. This sequence of verbs is often abbreviated with the acronym CRUD.

**Table 1-2.** CRUD Operations Summary

| Operation Type | HTTP Verb | Operation Summary |
|---|---|---|
| Create | POST | Creates a new object. |
| Retrieve | GET | Retrieves the representation of an existing object. |
| Update | PUT | Modifies an existing object. |
| Delete | DELETE | Deletes an existing object. |

## vCloud API REST Responses

All responses include an HTTP status code and, unless the status code is 204 (No Content), a Content-Type header. Response content depends on the request. Some responses include a document body, some include only a URL, and some are empty.

### Response Content

Response content depends on the requested operation. The response to a GET request is typically the complete representation of an existing object. The response to a PUT or POST request always contains values for the `href`, `name`, and `id` attributes of the object being created or updated. It also contains at most one `Task` element that you can retrieve to track the progress of the operation. When the `Task` completes with a status of `success`, a GET request to the object's `href` returns all properties of the object. If the `Task` completion status is not success, the object is in an indeterminate state, and should be deleted.

### HTTP Response Codes

A vCloud API client can expect a subset of HTTP status codes in a response.

**Table 1-3.** HTTP Status Codes that the vCloud API Returns

| Status Code | Status Description |
|---|---|
| 200 OK | The request is valid and was completed. The response includes a document body. |
| 201 Created | The request is valid. The requested object was created and can be found at the URL specified in the Location header. |
| 202 Accepted | The request is valid and a task was created to handle it. This response is usually accompanied by a `Task` element. |
| 204 No Content | The request is valid and was completed. The response does not include a body. |

**Table 1-3.** HTTP Status Codes that the vCloud API Returns (Continued)

| Status Code | Status Description |
|---|---|
| 400 Bad Request | The request body is malformed, incomplete, or otherwise invalid. |
| 401 Unauthorized | An authorization header was expected but not found. |
| 403 Forbidden | The requesting user does not have adequate privileges to access one or more objects specified in the request. |
| 404 Not Found | One or more objects specified in the request could not be found in the specified container. |
| 405 Method Not Allowed | The HTTP method specified in the request is not supported for this object. |
| 406 Not Acceptable | The resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request. |
| 409 Conflict | The object state is not compatible with the requested operation. |
| 415 Unsupported Media Type | The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method. |
| 500 Internal Server Error | The request was received but could not be completed because of an internal error at the server. |
| 504 Gateway Timeout | The server, while acting as a gateway or proxy, did not receive a timely response from the upstream server specified by the URI. |

# About the Schema Reference

The *vCloud API Schema Reference* includes reference material for all elements, types, queries, and operations in the vCloud API. It also includes the schema definition files.

The schema reference is available in HTML format in the vCloud Director documentation center.

# Setting Up for PHP Development 2

To use the vCloud SDK for PHP, you need PHP 5.3.2 or later and the PEAR HTTP_Request2 package or a similar HTTP client for PHP.

## Prerequisites for Using the vCloud SDK for PHP

To use the vCloud SDK for PHP, you should be familiar with the PHP programming language and have access to an installation of VMware vCloud Director.

In addition, consider the following items:

- The vCloud SDK for PHP 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 samples, you must use the vCloud 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 `CatalogAuthor` role should provide all of the necessary rights. For more information about roles and rights, see the *VMware vCloud Director Administrator's Guide*.
- Several of the sample programs, including `hellovcloud.php`, require you to have an OVF package available on the client host. This package must be uncompressed. For more information about OVF, see the *vCloud API Programming Guide*.

This chapter includes the following topics:



## Download the vCloud SDK for PHP Package

The vCloud SDK for PHP is distributed in two compressed archive formats. Uncompressed, either archive requires about 32 MB of disk space.

### Procedure

1. In a browser, go to http://www.vmware.com/go/vcloudsdkforphp.
2. In the Resources area of the vCloud SDK for PHP Community page, click the **Download** button.
3. On the Download page, log in with your VMware customer credentials.
4. Review the license agreement.

   Click **Yes** to accept it and continue with the download, or click **No** to exit without downloading.

5 On the Download page, choose a download option and click the file format to download.

| Option | Description |
|---|---|
| **vcloudPHP_5.1.0.*build*.tar.gz** | A compressed archive in `tar` format, where *build* is a build number. |
| **vcloudPHP_5.1.0.*build*.zip** | A compressed archive in `zip` format. |

6 When the download is complete, uncompress the download package into any convenient folder on your computer.

The package includes the following folders:

**docs** Reference documentation in HTML format.

**library** A collection of class libraries and functions that encapsulate vCloud API objects and operations.

**samples** Example code demonstrating common use cases associated with programmatically managing virtual infrastructure.

# Using the HTML Reference Material

The reference documentation in the `docs` folder of the vCloud SDK for PHP downloaded files provides detailed information about classes and functions in the SDK.

### Procedure

1 Open the `docs` folder in the downloaded files and open the `index.html` file in a browser.

2 Select **VMware_VCloud_API** from the **Packages** drop-down menu.

3 Select a class in the left-hand pane.

4 In the **Method Summary** section of the right-hand pane, click the link for the `__construct()` method.

The method summary lists the constructors for required and optional attributes, and elements of the class, sorted by type. You can click the name of any element, then click its method summary to view information about its constructors. For example, `VMware_VCloud_API_AdminOrgType` requires a `VMware_VCloud_API_OrgSettingsType` element. You can click the element name to see its method summary, and click its `__construct()` method to see how to construct it.

# Working with the vCloud SDK for PHP 3

The vCloud SDK for PHP provides a PHP class library 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.

## Packages Included in the vCloud SDK for PHP

The vCloud SDK for PHP includes the following packages:

**API packages**

API packages contain classes that represent complex types defined in vCloud API, vCloud administrative API, and vCloud vSphere platform API extensions. Classes in the API package are generated from the vCloud API XML schema files. Each class maps to a complex type defined in those files. Objects of these classes are referred to as vCloud data objects.

**Table 3-1.** VMware_VCloud_API Packages

| Package Name | Package Contents |
|---|---|
| VMware_VCloud_API | Classes representing objects defined in the vCloud user API and administrative API |
| VMware_VCloud_API_OVF, VMware_VCloud_API_OVFENV | Classes representing objects defined in the OVF specification |
| VMware_VCloud_API_Extension | Classes representing objects defined in the vCloud API extensions |
| VMware_VCloud_API_Version | Classes representing objects that contain vCloud API version information |

**SDK packages**

These packages contain classes that implement vCloud API operations. Each of the classes maps to a vCloud resource entity. Classes manage the resource entity life cycle of create, retrieve, update, and delete. This sequence of verbs is often abbreviated with the acronym CRUD. This package also implements utility functions associated with connecting to a vCloud instance, marshalling requests, unmarshalling responses, and so on. Objects of these classes are referred to as vCloud SDK objects.

**Table 3-2.** VMware_VCloud_SDK Packages

| Package Name | Package Contents |
|---|---|
| VMware_VCloud_SDK | Classes that implement operations defined in the vCloud user API and administrative API |
| VMware_VCloud_SDK_Extension | Classes that implement operations defined in the vCloud API vSphere Platform Extensions |
| VMware_VCloud_SDK_HTTP | Classes that support HTTP client operations. |

This chapter includes the following topics:



# Summary of SDK Objects, Containers, and Methods

Every SDK object is associated with a container type and an object reference creation method.

To create an SDK object, you use an SDK object creation method to retrieve an object reference from an object container. This table summarizes the types of SDK objects you can create and, for each object, lists the container object and the method for retrieving a reference from the container. The table omits the `VMware_VCloud_SDK_` part of the package names in the SDK Object and Container columns.

NOTE Rows where SDK Object is listed as None indicate operations that return a read-only object, such as a `RightReference`, that you might need when you create other objects.

**Table 3-3.** Summary of SDK Objects, Containers, and Methods

| SDK Object | Container | Method |
|---|---|---|
| None | Admin | getRightRefs() |
| ProviderVdc | Admin | getProviderVdcRefs() |
| None | Extension_VMWProviderVdc | getNetworkPoolRefs() |
| None | Extension_VimServer | getResourcePoolRefs() |
| Admin | None | See "Create a Root Object," on page 25 |
| AdminCatalog | AdminOrg | getAdminCatalogRefs() |
| AdminNetwork | AdminOrg | getAdminNetworkRefs() |
| AdminOrg | Admin | getAdminOrgRefs() , getSystemOrgRef() |
| AdminVdc | AdminOrg | getAdminVdcsRefs() |
| Catalog | Org | getCatalogRefs() |
| Task | Org | getTasks() |
| CatalogItem | Catalog | getCatalogItemRefs() |
| CatalogItem | AdminCatalog | getCatalogItemRefs() |

**Table 3-3.** Summary of SDK Objects, Containers, and Methods (Continued)

| SDK Object | Container | Method |
|---|---|---|
| Extension | None | See "Create a Root Object," on page 25 |
| Extension_Host | Extension | getHostRefs() |
| Extension_VimServer | Extension | getVimServerRefs() |
| Extension_VMWExternalNetwork | Extension | getVMWExternalNetworkRefs() |
| Extension_VMWNetworkPool | Extension | getVMWNetworkPoolRefs() |
| Extension_VMWProviderVdc | Extension | getVMWProviderVdcRefs() |
| Extension_BlockingTask | Extension | getBlockingTaskRefs() |
| Group | AdminOrg | getGroupRefs() |
| Media | Vdc | getMediaRefs() |
| Org | Service | getOrgRefs() |
| Role | Admin | getRoleRefs() |
| Service | None | See "Create a Root Object," on page 25 |
| User | AdminOrg | getUserRefs() |
| VApp | Vdc | getVAppRefs() |
| VApp | VApp | getContainedVAppRefs() |
| VAppTemplate | Vdc | getVAppTemplateRefs() |
| Vdc | Org | getVdcRefs() |
| Vm | VApp | getContainedVmRefs() |
| Extension_Datastore | Extension | getDatastoreRefs() |
| Disk | Vdc | getDiskRefs() |
| VdcStorageProfile | Vdc | getVdcStorageProfileRefs() |
| Right | Admin | getRightRefs() |
| AdminVdcStorageProfile | AdminVdc | getAdminVdcStorageProfileRefs() |
| ProviderVdcStorageProfile | AdminVdcStorageProfile | getProviderVdcStorageProfileRefs() |
| Extension_StrandedItem | Extension | getStrandedItems() |
| UserService | Service | getUserServiceRefs() |
| APIDefinition | UserService | getAPIDefinitionRefs() |
| TasksList | Org | getTasksListRef() |
| EdgeGateway | AdminVdc | getEdgeGatewayRefs() |
| Extension_VMWProviderVdcStorageProfile | Extension_VMWProviderVdc | getStorageProfileRefs() |
| Extension_VMWProviderVdcResourcePool | Extension_VMWProviderVdc | getResourcePoolRefs() |
| Extension_Service | Extension | getExtensionService() |
| Extension_ApiFilter | Extension_Service | getApiFilterRefs() |
| Extension_ServiceLink | Extension_Service | getServiceLinks() |
| Extension_APIDefinition | Extension_Service | getAPIDefinitions() |
| Extension_File | Extension_Service | getFileDescriptor() |

**Table 3-3.** Summary of SDK Objects, Containers, and Methods (Continued)

| SDK Object | Container | Method |
|---|---|---|
| Extension_ResourceClass | Extension_Service | getResourceClass() |
| Extension_ResourceClassAction | Extension_Service | getResourceClassAction() |
| Extension_AclRule | Extension_Service | getAclRule() |
| Extension_ServiceResource | Extension_Service | getServiceResources() |

# Create an SDK Object

To create an SDK object, retrieve an array of object references, and use the createSDKObj method to create an object from a reference.

You can create an SDK object when you need to run a life cycle operation such as create or modify on a vCloud API object. Most class constructors for SDK objects require two parameters:

- A `VMware_VCloud_SDK_Service` object, which contains HTTP connection information.
- A `ReferenceType` data object, which contains the request URL. For more information about request URLs, see "vCloud API REST Requests," on page 16.

For example, you can use code similar to the fragment shown in "Example: Creating an SDK Object," on page 24 to create a `VMware_VCloud_SDK_Org` object to use as an entry point for client operations. This procedure uses the data in Table 3-3 as a guide to creating SDK objects.

### Prerequisites

Familiarize yourself with the set of SDK objects, container objects, and constructor methods listed in Table 3-3. Examples in this procedure refer to column names in that table.

### Procedure

1 Retrieve an array of object references by specifying a container object and creation method.

```
$references=Container->Method
```

2 For any reference in the array, create an SDK object using the selected reference, as the following example shows.

```
SDK_Object=$service->createSDKObj($reference)
```

## Example: Creating an SDK Object

```
// get the list of all organizations in the vCloud
$orgRefs = $service->getOrgRefs($orgName);
// create an object that represents the first organization in the list
$sdkOrg = $service->createSDKObj($orgRefs[0]);
// create a task object
$sdkTask = $service->createSDKObj($task);
```

NOTE Several new SDK objects have specialized creation methods. The following example creates a QueryService SDK object:

```
$sdkQuery= VMware_VCloud_SDK_Query::getInstance($service);
```

# Create a Data Object

To create a data object, you can either invoke an empty constructor and then call the setters for the object or invoke the constructor with parameters.

Each data object class includes a constructor method whose parameters represent attributes of the class and all of its ancestors. Attributes are marked as protected to restrict their visibility. All classes contain setter and getter methods for XML elements and attributes.

The general form of setter and getter method names is *operation_attribute-name* for attributes and *operationElementName* for elements, where *operation* is one of `set` or `get`. For example, the `VMware_VCloud_API_ReferenceType` class supports `set_name()` and `get_name()` methods that get or set the value of its `name` attribute. The `VMware_VCloud_API_UserType` class supports `setFullName()` and `getFullName()` methods that set or get the value of the `FullName` element in a `User` object.

### Procedure

- Create a data object by invoking an empty constructor and calling the setters for the object.

```
$ref = new VMware_VCloud_API_ReferenceType();
   $ref->set_href($href);
   $ref->set_type($type);
   $ref->set_name($name);
```

- Create a data object by invoking the constructor with parameters.

```
$ref = new VMware_VCloud_API_ReferenceType ($href=$href, $type=$type, $name=$name);
```

# Create a Root Object

In the vCloud API, root objects such as `VCloud` and `VMWExtension` do not have containers. The vCloud SDK for PHP provides dedicated constructors for these objects.

### Procedure

- To create a `VMware_VCloud_SDK_Service` object to use as an entry point for user API operations, use `VMware_VCloud_SDK_Service::getService`, as this example shows:

```
$service = VMware_VCloud_SDK_Service::getService();
```

- To create a VMware_VCloud_SDK_Admin object to use as an entry point for administrative operations, use `createSDKAdminObj`, as this example shows:

```
$sdkAdminObj = $service->createSDKAdminObj();
```

- To create a `VMware_VCloud_SDK_Extension` object to use as an entry point for vSphere Platform Extensions operations, use `createSDKExtensionObj`, as this example shows:

```
$sdkExtObj = $service->createSDKExtensionObj();
```

# Use a Different HTTP Library

Example programs included in the vCloud SDK for PHP require the PEAR HTTP_Request2 package. You can use a different HTTP library.

### Procedure

1 Create an HTTP client object that implements the `VMware_VCloud_SDK_Http_Client_Interface` interface.

2 Call the `VMware_VCloud_SDK_Service::getService()` method that specifies the client that you created.

For example,

```
$service = VMware_VCloud_SDK_Service::getService($myHTTPClient);
```

.

# About the Example Programs

4

The vCloud SDK for PHP includes example programs that demonstrate how to use the SDK to develop client applications. The examples are in the `samples` folder of the SDK downloadable files.

Comments in the examples provide detailed information about how they use the features of the vCloud SDK for PHP.

## Required Permissions

Some of the example programs require system administrator privileges to run. Others can be run by any user who can create and operate a vApp.

**Table 4-1.** Summary of Example Programs and Required Permissions

| Example Name | Description | Required Permissions |
|---|---|---|
| `login.php` | Authenticates a user. | Requires credentials for a system administrator or user with the vApp Author role. |
| `ssologin.php` | Authenticates a user using a SAML identity provider (including the vSphere SSO server). | Requires the SAML token provided by the system's SSO server or SAML identity provider. |
| `createcatalogitem.php` | Adds an item to a catalog. | vApp Author. |
| `createcatalog.php` | Creates a catalog. | vApp Author. |
| `deployvapp.php` | Deploys and powers on a vApp. | vApp Author. |
| `hellovcloud.php` | A structured workflow example that uses command-line parameters. | vApp Author. |
| `instantiatevapptemplate.php` | Instantiates a vApp template using organization defaults. | vApp Author. |
| `updatevm.php` | Edits the memory required by a virtual machine and reduces the existing value by half. | vApp Author. |
| `uploadvapptemplate.php` | Uploads an OVF package to create a vApp template. | vApp Author. |
| `createorg.php` | Creates an organization. | System Administrator. |
| `createvdc.php` | Creates a vDC. | System Administrator. |
| `createextnet.php` | Creates an external network from vSphere resources. | System Administrator. |
| `createnetpool.php` | Creates a network pool from vSphere resources. | System Administrator. |

**Table 4-1.** Summary of Example Programs and Required Permissions (Continued)

| Example Name | Description | Required Permissions |
|---|---|---|
| `createprovidervdc.php` | Creates a provider vDC from vSphere resources. | System Administrator. |
| `importvm.php` | Imports a virtual machine from vCenter to create a vApp in the specified vDC. | System Administrator. |
| `host.php` | Prepares an ESX/ESXi host for use with vCloud Director. | System Administrator. |
| `vimserver.php` | Register, unregister, enable, or disable a vCenter server for use with vCloud Director. | System Administrator. |
| `recomposevapp.php` | Add a virtual machine from a vAppTemplate to an existing vApp. | System Administrator. |
| `query.php` | Use the query service. | System Administrator. |
| `callout.php` | Use notifications and blocking tasks. | System Administrator. |
| `enableblockingtasks.php` | Enable or disable blocking tasks. | System Administrator. |
| `edgegatewaycrud.php` | Create, retrieve, update, or delete an edge gateway. | System Administrator. |
| `externalnetworkcrud.php` | Create, retrieve, update, or delete an external network. | System Administrator. |
| `networkpoolcrud.php` | Create, retrieve, update, or delete a network pool. | System Administrator. |
| `providervdccrud.php` | Create, retrieve, update, or delete a provider vDC. | System Administrator. |

This chapter includes the following topics:



# Running the HellovCloud Example

The `hellovcloud.php` example program, included in the `samples` folder of the SDK, demonstrates a number of the operations that the vCloud SDK for PHP supports.

The `hellovcloud.php` example demonstrates the following operations:

- Logging in to the cloud and getting an organization list
- Finding a vDC and a catalog
- Uploading an OVF package to create a vApp template in the catalog
- Instantiating the vApp template to create a vApp
- Operating the vApp

Like all of the example programs in this SDK, `hellovcloud.php` is liberally commented. Read the comments for more information about how this example uses the features of the SDK to implement a structured workflow through the lifecycle of a vApp.

### Procedure

1 Open a console or shell in the `samples` folder.

2 Run `php hellovcloud.php`, as show in "Example: Running HellovCloud," on page 29.

## Example: Running HellovCloud

You must supply runtime options on the command line. To see a summary of `hellovcloud.php` options, use the following command:

```
php hellovcloud.php --help
```

To run the `hellovcloud.php` example, use the following command:

```
php hellovcloud.php -s vCloudURL -u user@vcloud-organization -p password -o=orgName -d=vdcName -
v=ovfFileLocation -g=catalogName -n=vAppTemplateName
```

Values for the runtime options provide user credentials, object names, and file names.

**Table 4-2.** hellovcloud runtime options

| Value | Description |
|---|---|
| *vCloudURL* | The vCloud Director URL. |
| *user* | The name of a user account that can run the sample. This user must have rights to create and operate vApps. |
| *vcloud-organization* | The name of the organization in which the user account exists. |
| *password* | The user's password. |
| *orgName* | The name of the organization in which the user account exists, and to which the user is authenticating. |
| *vdcName* | The name of a vDC in that organization where the user can upload the OVF and deploy the vApp. |
| *ovfFileLocation* | The full pathname to the OVF descriptor on the local disk. |
| *catalogName* | The name of the catalog into which the OVF package is uploaded, and in which the resulting vApp template is catalogued. |
| *vAppTemplateName* | The name of the vAppTemplate to be created from the OVF package. |

All options but `-s`, `-u`, and `-p` must be separated from their arguments by an equals sign, as the following example shows:

```
php hellovcloud.php -s https://vcloud.example.com -u user@SampleOrg -p Pa55w0rd -o=SampleOrg
-d=SampleVdc -v=C:\descriptor.ovf -g SampleCatalog -n=SamplevAppTemplate
```

You can use the vCloud Director Web Console or the vCloud REST API to find appropriate values in your vCloud for *orgName*, *vdcName*, and *catalogName*. See the *vCloud Director User's Guide* or the *vCloud API Programming Guide*.

# Understanding the HellovCloud Example

The `hellovcloud.php` example performs a sequence of operations that are typical of the workflow for provisioning and operating a vApp.

Included in the example are comment blocks that explain how the steps in the example use the SDK libraries. For additional information about vCloud API requests, see the *vCloud API Programming Guide*.

### 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 that Basic HTTP authentication requires. This form is MIME Base64 encoding of a string having the form *user@vcloud-organization*:*password*. The `VMware_VCloud_SDK_Service` class implements a login method that takes the following parameters:

| | |
|---|---|
| **`userName`** | Supplied in the form *user@vcloud-organization*. |
| **`password`** | The user's password. |

`hellovcloud.php` encapsulates this authentication protocol in its `login()` method, which returns a list of organizations to which the specified user has access. In the typical case, this list has a single member, the organization that was supplied in the `userName` parameter.

### Getting References to the vDC and Catalog

To instantiate a vApp template and operate the resulting vApp, you need the object references (the `href` values) for the catalog in which the vApp template will be entered and the vDC in which the vApp will be deployed. The `VMware_VCloud_SDK_Org` class implements `getVdcRefs()` and `getCatalogRefs()` methods that return references to vDCs and catalogs.

### Creating a vApp Template in the Catalog by Uploading an OVF Package

The `hellovcloud.php` command requires you to supply the name of an OVF descriptor file. This information is used in the `uploadOVFAsVappTemplate` method to upload the OVF descriptor file and create a vApp template.

### Creating a vApp by Instantiating the vApp Template

After the template is added to a catalog, you can create a vApp by instantiating the template. `hellovcloud.php` implements an `instantiateVAppTemplateDefault()` that constructs a simple `InstantiateVAppTemplateParams` request body, makes the request to the `action/instantiateVAppTemplate` URL of the vDC, and returns a helper object that contains a reference to the vApp.

### Operating the vApp

The `VMware_VCloud_SDK_VApp` class includes methods that perform operations on the vApp. Most of these operations return a `Task` object that tracks the progress of the operation. `hellovcloud.php` uses these methods to cycle the vApp through the following states:

1 `$sdkVApp->deploy()`, which deploys and powers on the vApp

2 `$sdkVApp->undeploy()`, which powers off and undeploys the vApp

3 `$sdkVApp->delete()`, which deletes the vApp

## Run the Other Example Programs

The example programs included with the SDK display a usage message when you run them with no parameters.

Each of the example programs in the `samples` folder requires that you specify parameters on the command line. Common parameters, such as the credentials that the examples use for logging in, are read from a file named `config.php`. Running an example program with no command-line parameters causes the program to display a usage summary. You can use the summary to help construct a command line that runs the example with parameters that are appropriate for your installation.

### Procedure

1 (Optional) Edit the `config.php` file to provide common parameter values.

When you run any example program, you can override its use of these values by supplying them on the command line.

2 Run the example in a shell window using a command of the following form, where *example* is the name of the example program:

```
php example.php
```

When you run an example program with no parameters, it displays a usage message and exits.

# Index