

# vCloud SDK for PHP Developer's Guide

vCloud Director 1.5

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-000598-00

**vmware**<sup>®</sup>

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](mailto:docfeedback@vmware.com)

Copyright © 2010, 2011 VMware, Inc. All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. VMware products are covered by one or more patents listed at <http://www.vmware.com/go/patents>.

VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.

**VMware, Inc.**  
3401 Hillview Ave.  
Palo Alto, CA 94304  
[www.vmware.com](http://www.vmware.com)

# Contents

vCloud SDK for PHP Developer's Guide	5
<b>1 About the VMware vCloud API</b>	<b>7</b>
Object Taxonomy	8
Objects, References, and Representations	9
Links and Link Relations	10
Client Workflow Overview	13
About the Schema Reference Downloadable Archive	15
<b>2 Setting Up for PHP Development</b>	<b>19</b>
Download the vCloud SDK for PHP Package	19
Using the HTML Reference Material	20
<b>3 Working with the vCloud SDK for PHP</b>	<b>21</b>
Summary of SDK Objects, Containers, and Methods	22
Create an SDK Object	23
Create a Data Object	24
Create a Root Object	25
Use a Different HTTP Library	25
<b>4 About the Example Programs</b>	<b>27</b>
Run the Hello vCloud Example	28
Run the Other Example Programs	29
Index	31



# vCloud SDK for PHP Developer's Guide

---

The *vCloud SDK for PHP Developer's Guide* provides information about the PHP SDK for version 1.5 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
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

---

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.

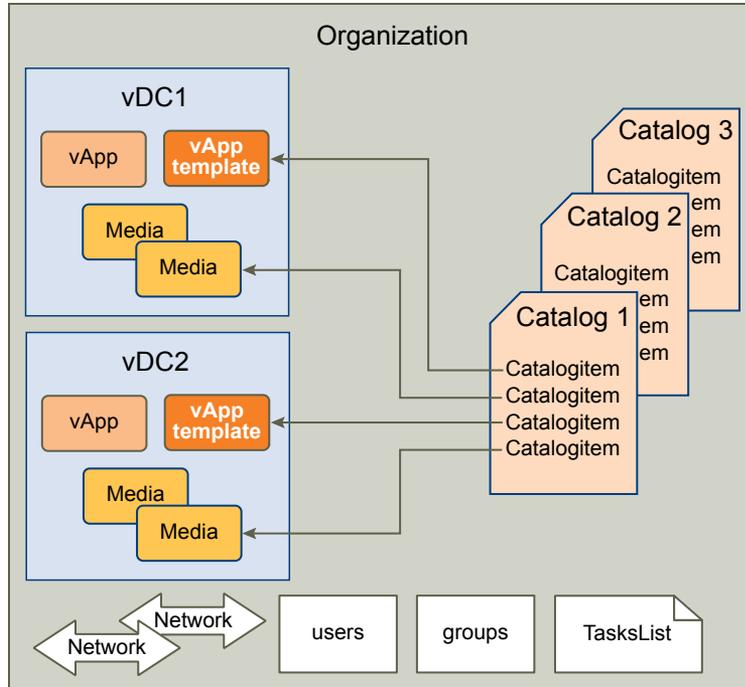
This chapter includes the following topics:

- [“Object Taxonomy,”](#) on page 8
- [“Objects, References, and Representations,”](#) on page 9
- [“Links and Link Relations,”](#) on page 10
- [“Client Workflow Overview,”](#) on page 13
- [“About the Schema Reference Downloadable Archive,”](#) on page 15

## 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.

### 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. 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.

### 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.

### Networks

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

**Virtual Datacenters**

A virtual datacenter (vDC) is a deployment environment for virtual systems and 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.

**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:

- 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.

<b>id</b>	The object identifier, expressed in URN format. The value of the <code>id</code> attribute uniquely identifies the object, persists for the life of the object, and is never reused. The <code>id</code> attribute value is intended to provide a context-free identifier that can be used with the vCloud API <code>entityResolver</code> and is also suitable for use by clients that need to access the object using a different API.
<b>type</b>	The object type, specified as a MIME content type.
<b>href</b>	An object reference, expressed in URL format. Because this URL includes the object identifier portion of the <code>id</code> attribute value, it uniquely identifies the object, persists for the life of the object, and is never reused. The value of the <code>href</code> 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 <code>href</code> as an opaque string. The rules that govern how the server constructs <code>href</code> 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:

<b>rel</b>	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 <code>href</code> value as a request URL.
<b>type</b>	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.
<b>href</b>	An object reference, expressed in URL format. Because this URL includes the object identifier portion of the <code>id</code> attribute value, it uniquely identifies the object, persists for the life of the object, and is never reused. The value of the <code>href</code> 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 <code>href</code> as an opaque string. The rules that govern how the server constructs <code>href</code> strings might change in future releases.
<b>name</b>	The name of the referenced object, taken from the value of that object's <code>name</code> attribute. Action links do not include a <code>name</code> attribute.

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

<b>rel Attribute Value</b>	<b>Action or Relationship Description</b>	<b>Implied HTTP Verb</b>
add	Add an item to this container.	POST
alternate	References an alternate representation of this object.	GET
catalogItem	References the <code>CatalogItem</code> object that refers to this object.	GET
collaboration:abort	Abort this blocking task.	POST
collaboration:fail	Fail this blocking task.	POST
collaboration:resume	Resume this blocking task.	POST
consolidate	Consolidate this virtual machine.	POST
controlAccess	Apply access controls.	POST
copy	Reserved, unimplemented.	N/A
deploy	Deploy this vApp.	POST
disable	Disable this object.	POST
discardState	Discard the suspended state of this virtual machine.	POST
down	References an object contained by this object.	GET
download:alternate	Reserved, unimplemented.	N/A
download:default	References the default location from which this file can be downloaded.	GET
edit	Modify this object.	PUT
enable	Enable this object.	POST
firstPage	Reference to the first page of a paginated response.	GET
installVmwareTools	Install VMware Tools on this virtual machine.	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
move	Reserved, unimplemented.	N/A
nextPage	Reference to the next page of a paginated response.	GET
ova	Reserved, unimplemented	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

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

<b>rel Attribute Value</b>	<b>Action or Relationship Description</b>	<b>Implied HTTP Verb</b>
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.	POST
reconnect	Reconnect this vCenter Server to this cloud.	POST
register	Register a VCenter Server to this cloud.	POST
reject	Reject this request.	POST
relocate	Relocate this virtual machine.	POST
remove	Remove this object.	DELETE
repair	Repair this ESX/ESXi host.	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
task:cancel	Cancel this task.	POST
blockingTask	A list of pending blocking task requests in this cloud.	GET
taskOwner	Reference to the owner of a task	GET
taskParams	Reference to the request parameters of a task	GET
taskRequest	Reference to the request associated with a task	GET
undeploy	Undeploy this vApp.	POST
unlock	Unlock a user account	POST
unregister	Unregister this vCenter Server.	POST
up	References an object that contains this object.	GET
updateProgress	Request an update of this task's progress.	POST
upgrade	Upgrade this ESX/ESXi host.	POST
upload:alternate	Reserved, unimplemented.	N/A
upload:default	References the default location to which this object can be uploaded.	PUT

## 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, documents that represent 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:

- What is the set of objects that the API supports, and what do they represent. For example, what is a vDC and how does it relate to an organization or catalog?
- How does 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 does the client refer 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 and 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 Downloadable Archive,”](#) on page 15.

### 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 a 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 list of vCloud API versions that the server supports. Each version has its own 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. See .

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.

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.
303 See Other	The response to the request can be found at the URL specified in the Location header.
400 Bad Request	The request body is malformed, incomplete, or otherwise invalid.

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

Status Code	Status Description
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.
500 Internal Server Error	The request was received but could not be completed because of an internal error at the server.
501 Not Implemented	The server does not implement the request.
503 Service Unavailable	One or more services needed to complete the request are not available on the server.

## About the Schema Reference Downloadable Archive

XML schema reference documentation in HTML format for the vCloud API is available as a downloadable archive. This archive also includes the schema definition files, and examples XML representations of vCloud API objects.

To use the reference documentation:

- 1 Download the compressed archive from <http://www.vmware.com/support/vcd/doc/rest-api-doc-1.5-html.zip>
- 2 Uncompress the archive into any convenient folder.
- 3 In the folder, open the file `index.html` in a browser.

## How the Schema Reference Documentation is Organized

The schema reference documentation is organized to reflect the division of the vCloud API into user, administrator, and extension categories. Within each category, you can open a list of elements, types that the elements extend, and operations that create, retrieve, update, or delete the objects that the elements represent.

### **User Operations, Elements, and Types**

These operations are performed by all users who have permission to log into an organization. User elements and user types represent the objects that these operations manipulate.

### **Administrator Operations, Elements, and Types**

These operations are performed by organization administrators or system administrators. Administrator elements and types represent the objects that these operations manipulate.

### **Extension Operations, Elements, and Types**

These operations are performed by system administrators who need access to vSphere platform objects from the vCloud API. Extension elements and types represent the objects that these operations manipulate.

## Searching In a Category

You can enter a search string in the Quick Index text box to search the lists of operations, elements, and types in any category.

- In an **Operations** list, you can search for the following items:
  - All or part of the name of the object on which you want to operate. The search returns a list of all of the operations that are possible on that object. For example, selecting User Operations and typing **vApp** in the Quick Index text box returns a list of all of the requests that operate on a vApp object.
  - The name of an action to perform. For example, selecting User Operations and typing **power** in the Quick Index text box returns a list of all the requests that change the power state of a vApp.
  - An HTTP verb (GET, PUT, POST, DELETE) to view a list of all the requests that use that verb. For example, selecting User Operations and typing **PUT** in the Quick Index text box returns a list of all of the requests that update an object.
- In an **Elements** or **Types** list, type all or part of the element or type name.

Search terms are not case-sensitive.

## Operation Summary Syntax

Operations consist of an HTTP verb and a request URL. The reference documentation represents the verb and the URL using the following syntax:

```
HTTP_VERB /object_type/{id}[/action/action_name]
```

In this syntax, the initial / character is assumed to follow a site-specific API URL, such as `https://vcloud.example.com/api`. The following strings represent variables in the remainder of the URL:

<b>HTTP_VERB</b>	The HTTP verb used to request the operation.
<b>object_type</b>	An abbreviation of the MIME type of the object referenced by the operation. This abbreviation is constructed from the final component of the object's media type, between the . and the +xml designation. For example, for an object whose media type is <code>application/vnd.vmware.vcloud.catalogItem+xml</code> , the <i>object_type</i> is shown as <code>catalogItem</code> .
<b>{id}</b>	The unique identifier of the object of the operation.
<b>action_name</b>	The name of an action. Required only when the operation request URL includes the string <code>/action/</code> .

## Element and Type Reference Pages

For each element or complex type, the reference documentation provides a page that lists the following items:

<b>Element</b>	The name of the element.
<b>Type</b>	The name of the type that the element extends.
<b>Namespace</b>	The XML namespace in which this element or type name is defined.
<b>Description</b>	A description of the purpose and contents of the element or type.
<b>Since</b>	The vCloud API version in which this element or type first appeared.
<b>Schema</b>	The name of the XML schema definition file in which this element or type is defined. Click to open the file in your browser, or right-click to download it.

<b>Media Type</b>	The MIME type associated with this element or type.																		
<b>Extends</b>	The base type from which this element is derived.																		
<b>XML Representation</b>	The XML representation of the element or type. Names of contained elements are links to the reference pages for those elements.																		
<b>Attributes</b>	A table listing the following properties of each attribute of the element or type: <table> <tr> <td><b>Attribute</b></td> <td>The name of the attribute.</td> </tr> <tr> <td><b>Type</b></td> <td>The primitive XML type of the attribute.</td> </tr> <tr> <td><b>Required</b></td> <td>Yes for attributes that are required. No for attributes that are optional.</td> </tr> <tr> <td><b>Modifiable</b></td> <td>A value of <code>always</code> means that a client request can modify the value of this attribute. A value of <code>create</code> means that this attribute can be set or modified only as part of object creation. A value of <code>none</code> means that this attribute is read-only.</td> </tr> <tr> <td><b>Since</b></td> <td>The vCloud API version in which this attribute first appeared.</td> </tr> <tr> <td><b>Description</b></td> <td>A description of the purpose and contents of the attribute.</td> </tr> </table>	<b>Attribute</b>	The name of the attribute.	<b>Type</b>	The primitive XML type of the attribute.	<b>Required</b>	Yes for attributes that are required. No for attributes that are optional.	<b>Modifiable</b>	A value of <code>always</code> means that a client request can modify the value of this attribute. A value of <code>create</code> means that this attribute can be set or modified only as part of object creation. A value of <code>none</code> means that this attribute is read-only.	<b>Since</b>	The vCloud API version in which this attribute first appeared.	<b>Description</b>	A description of the purpose and contents of the attribute.						
<b>Attribute</b>	The name of the attribute.																		
<b>Type</b>	The primitive XML type of the attribute.																		
<b>Required</b>	Yes for attributes that are required. No for attributes that are optional.																		
<b>Modifiable</b>	A value of <code>always</code> means that a client request can modify the value of this attribute. A value of <code>create</code> means that this attribute can be set or modified only as part of object creation. A value of <code>none</code> means that this attribute is read-only.																		
<b>Since</b>	The vCloud API version in which this attribute first appeared.																		
<b>Description</b>	A description of the purpose and contents of the attribute.																		
<b>Elements</b>	A table listing the following properties of each element defined in the type: <table> <tr> <td><b>Element</b></td> <td>The name of the element.</td> </tr> <tr> <td><b>Type</b></td> <td>A link to the definition of the complex type that the element is based on.</td> </tr> <tr> <td><b>Occurrence</b></td> <td>The occurrence constraint for the element. The constraint can be one of the following expressions: <table> <tr> <td><b>0..*</b></td> <td>Optional. Can occur zero or more times.</td> </tr> <tr> <td><b>0..1</b></td> <td>Optional. Can occur at most once.</td> </tr> <tr> <td><b>1</b></td> <td>Required. Must occur exactly once.</td> </tr> </table> </td> </tr> <tr> <td><b>Modifiable</b></td> <td>A value of <code>always</code> means that a client request can modify the contents of this element. A value of <code>create</code> means that element contents can be set or modified only as part of object creation. A value of <code>none</code> means that this element is read-only.</td> </tr> <tr> <td><b>Since</b></td> <td>The vCloud API version in which this element first appeared.</td> </tr> <tr> <td><b>Description</b></td> <td>A description of the purpose and contents of the element.</td> </tr> </table>	<b>Element</b>	The name of the element.	<b>Type</b>	A link to the definition of the complex type that the element is based on.	<b>Occurrence</b>	The occurrence constraint for the element. The constraint can be one of the following expressions: <table> <tr> <td><b>0..*</b></td> <td>Optional. Can occur zero or more times.</td> </tr> <tr> <td><b>0..1</b></td> <td>Optional. Can occur at most once.</td> </tr> <tr> <td><b>1</b></td> <td>Required. Must occur exactly once.</td> </tr> </table>	<b>0..*</b>	Optional. Can occur zero or more times.	<b>0..1</b>	Optional. Can occur at most once.	<b>1</b>	Required. Must occur exactly once.	<b>Modifiable</b>	A value of <code>always</code> means that a client request can modify the contents of this element. A value of <code>create</code> means that element contents can be set or modified only as part of object creation. A value of <code>none</code> means that this element is read-only.	<b>Since</b>	The vCloud API version in which this element first appeared.	<b>Description</b>	A description of the purpose and contents of the element.
<b>Element</b>	The name of the element.																		
<b>Type</b>	A link to the definition of the complex type that the element is based on.																		
<b>Occurrence</b>	The occurrence constraint for the element. The constraint can be one of the following expressions: <table> <tr> <td><b>0..*</b></td> <td>Optional. Can occur zero or more times.</td> </tr> <tr> <td><b>0..1</b></td> <td>Optional. Can occur at most once.</td> </tr> <tr> <td><b>1</b></td> <td>Required. Must occur exactly once.</td> </tr> </table>	<b>0..*</b>	Optional. Can occur zero or more times.	<b>0..1</b>	Optional. Can occur at most once.	<b>1</b>	Required. Must occur exactly once.												
<b>0..*</b>	Optional. Can occur zero or more times.																		
<b>0..1</b>	Optional. Can occur at most once.																		
<b>1</b>	Required. Must occur exactly once.																		
<b>Modifiable</b>	A value of <code>always</code> means that a client request can modify the contents of this element. A value of <code>create</code> means that element contents can be set or modified only as part of object creation. A value of <code>none</code> means that this element is read-only.																		
<b>Since</b>	The vCloud API version in which this element first appeared.																		
<b>Description</b>	A description of the purpose and contents of the element.																		
<b>Operations</b>	A summary of the operations permitted on the element. Operations are categorized by request type; one of create, retrieve, update, and delete. This sequence of verbs is often abbreviated with the acronym CRUD.																		

## Schema Definition Files

XML schema definition files (\*.xsd) are included in the etc folder of schema reference downloadable archive. This folder contains several subfolders:

- 1.0** Schema definition files for vCloud API version 1.0.
- 1.5** Schema definition files for vCloud API version 1.5.
- schemas** Additional schema definition files that are version-independent or from external sources such as DMTF.

# Setting Up for PHP Development

---

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, and must specify a single `vmrk` file. For more information about OVF, see the *vCloud API Programming Guide*.

This chapter includes the following topics:

- “[Download the vCloud SDK for PHP Package](#),” on page 19
- “[Using the HTML Reference Material](#),” on page 20

## 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
<b>vcloudPHP_1.5.0.build.tar.gz</b>	A compressed archive in tar format, where <i>build</i> is a build number.
<b>vcloudPHP_1.5.0.build.zip</b>	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:

<b>docs</b>	Reference documentation in HTML format.
<b>library</b>	A collection of class libraries and functions that encapsulate vCloud API objects and operations.
<b>samples</b>	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

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,”](#) on page 22
- [“Create an SDK Object,”](#) on page 23
- [“Create a Data Object,”](#) on page 24
- [“Create a Root Object,”](#) on page 25
- [“Use a Different HTTP Library,”](#) on page 25

## 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 <a href="#">“Create a Root Object,”</a> 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 “ <a href="#">Create a Root Object</a> ,” 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()
Network	Org	getOrgNetworkRefs()
Org	Service	getOrgRefs()
Role	Admin	getRoleRefs()
Service	None	See “ <a href="#">Create a Root Object</a> ,” 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()

## 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 14.

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

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
<code>login.php</code>	Authenticates a user.	Requires credentials for a system administrator or user with the vApp Author role.
<code>createcatalogitem.php</code>	Adds an item to a catalog.	vApp Author.
<code>createcatalog.php</code>	Creates a catalog.	vApp Author.
<code>deployvapp.php</code>	Deploys and powers on a vApp.	vApp Author.
<code>hellovcloud.php</code>	A structured workflow example that uses command-line parameters.	vApp Author.
<code>instantiatevapptemplate.php</code>	Instantiates a vApp template using organization defaults.	vApp Author.
<code>updatevm.php</code>	Edits the memory required by a virtual machine and reduces the existing value by half.	vApp Author.
<code>uploadvapptemplate.php</code>	Uploads an OVF package to create a vApp template.	vApp Author.
<code>createorgnetwork.php</code>	Adds a network to an organization.	System Administrator.
<code>createorg.php</code>	Creates an organization.	System Administrator.
<code>createvdc.php</code>	Creates a vDC.	System Administrator.
<code>createextnet.php</code>	Creates an external network from vSphere resources.	System Administrator.
<code>createnetpool.php</code>	Creates a network pool from vSphere resources.	System Administrator.
<code>createprovidervdc.php</code>	Creates a provider vDC from vSphere resources.	System Administrator.

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

Example Name	Description	Required Permissions
<code>importvm.php</code>	Imports a virtual machine from vCenter to create a vApp in the specified vDC.	System Administrator.
<code>host.php</code>	Prepares an ESX/ESXi host for use with vCloud Director.	System Administrator.
<code>vimserver.php</code>	Register, unregister, enable, or disable a vCenter server for use with vCloud Director.	System Administrator.
<code>recomposevapp.php</code>	Add a virtual machine from a vAppTemplate to an existing vApp.	System Administrator.
<code>query.php</code>	Use the query service.	System Administrator.
<code>callout.php</code>	Use notifications and blocking tasks.	System Administrator.
<code>enableblockingtasks.php</code>	Enable or disable blocking tasks.	System Administrator.

This chapter includes the following topics:

- [“Run the Hello vCloud Example,”](#) on page 28
- [“Run the Other Example Programs,”](#) on page 29

## Run the Hello vCloud Example

The `hellovcloud.php` example program, included in the `samples` folder of the SDK, uses the vCloud SDK for PHP to implement a structured workflow through the lifecycle of a vApp.

The `hellovcloud.php` example program demonstrates these operations that the vCloud SDK for PHP supports:

- Logging in to a vCloud organization
- Browsing the organization to find a vDC and a catalog
- Instantiating a vApp template from the catalog to create a vApp
- Operating the vApp
- Logging out

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 vCloud SDK for PHP.

## Runtime Options

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 server -u user@orgName -p password -c config -o=orgName -d=vdcName
-g=catalogName -i=item -a=vAppName
```

The following options are required:

- *server* is the hostname or IP address of a vCloud Director server.
- *user* is the name of a vCloud Director user, in the form *user@orgName*, where *orgName* is the name of the organization to which the user is authenticating. This user must have rights to create and operate vApps.

- *password* is the user's password.

The following options are optional:

- *config* is a set of HTTPS connection parameters in the form of a PHP array. If you omit this option, `helloworld.php` accepts any server certificate. The following specification of *config* enables certificate validation, using a certificate stored in `/tmp/cert.crt`:  

```
-c='ssl_verify_peer=>true, ssl_verify_host=>true, ssl_cafire=>/tmp/cert.crt'
```
- *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 instantiate and deploy the vApp.
- *catalogName* is the name of a catalog in the user's organization.
- *item* is the value of the name attribute of a `CatalogItem` element that references the vApp template you plan to instantiate. This `CatalogItem` must be contained by the catalog specified by *catalogName*.
- *vAppName* is the name to give to the vApp that this example creates.

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

```
php helloworld.php -s vcloud.example.com -u user@exampleOrg -p Pa55w0rd -o=exampleOrg
-d=exampleVdc -g=exampleCatalog -i=exampleTemplate -a=MyVapp
```

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

## 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

## C

containers, SDK **22**

## D

data object, to create **24**

## E

Entity, object representation in **9**

example programs

    Hello vCloud **28**

    to run **29**

    using **27**

## H

HTTP library **25**

## I

id attribute **9**

## L

Link element, rel attribute **10**

## M

methods, SDK **22**

## O

object hierarchy, diagram of **8**

object identifiers **9**

object references, about **9**

objects

    root **25**

    SDK **22**

## P

PHP, supported versions **19**

PHP method summary **20**

PHP SDK, about **21**

## R

requests, about **14**

responses, about **14**

## S

schema files, accessing **15**

schema reference **15**

SDK, to download **19**

SDK object, to create **23**

## V

vCloud API, and RESTful programming style **7**

## W

workflow **13**

## X

XML

    compressed responses **14**

    validation of **14**

