2.2 CloudStack API Developer Guide

March 11th, 2011

© 2011 Cloud.com, Inc. All rights reserved. Specifications are subject to change without notice. The Cloud.com logo, Cloud.com, CloudStack, Hypervisor Attached Storage, HAS, Hypervisor Aware Network, HAN, and VMSync are trademarks or registered trademarks of Cloud.com, Inc. All other brands or products are trademarks or registered trademarks of their respective holders.

Contents

[1 Introduction 4](#_Toc287862202)

[1.1 Getting Started 4](#_Toc287862203)

[2 Making API Requests 5](#_Toc287862204)

[2.1 Signing API Requests 5](#_Toc287862205)

[3 Response Formats 8](#_Toc287862206)

[3.1 Error Handling 8](#_Toc287862207)

[4 Asynchronous Commands 9](#_Toc287862208)

[4.1 Job Status 9](#_Toc287862209)

[4.2 Example 9](#_Toc287862210)

[5 What’s New? 12](#_Toc287862211)

[5.1 What’s New in 2.2.x? 12](#_Toc287862212)

# Introduction

The CloudStack™ Web Services Query HTTP API is loosely based off of the REST architecture and allows developers to both create new management solutions or to integrate their existing systems with CloudStack. It supports POST/GET requests and returns both XML and JSON response formats. The CloudStack API supports three access roles.

* **Global Admin**. Access to all features of the cloud, including both virtual and physical resource management. You can access the 2.2. API via the link : <http://download.cloud.com/releases/2.2.0/api/TOC_Global_Admin.html>
* **Domain-Admin**. Access to only the virtual resources of the clouds that belong to the administrator’s domain. You can access the 2.2 API via the link :  <http://download.cloud.com/releases/2.2.0/api/TOC_Domain_Admin.html>
* **User**. Access to only the features that allow management of the user’s virtual instances, storage, and network. You can access the 2.2 API via the link : <http://download.cloud.com/releases/2.2.0/api/TOC_User.html>

The following document is meant to guide you on how to develop and access the CloudStack API.

## Getting Started

To get started on development against the API, you should have the following:

* URL to the CloudStack server you wish to integrate with.
* Both the API Key and Secret Key for an account. This should have been generated by the administrator of the cloud instance and given to you.
* Knowledge of how HTTP GET/POST and query strings work.
* Knowledge of either XML or JSON.
* Knowledge of a programming language that can generate HTTP requests (i.e. Java, PHP, etc.).

# Making API Requests

All CloudStack API requests are submitted in the form of a HTTP GET/POST with an associated command and any parameters. A request is composed of the following whether in HTTP or HTTPS:

* CloudStack API URL: This is the web services API entry point (i.e. <http://www.cloud.com:8080/client/api>)
* Command: The web services command you wish to execute (i.e. Start a virtual machine, create a disk volume).
* Parameters: Any additional required or optional parameters for the command

A sample API GET request looks like the following:

http://localhost:8080/client/api?command=deployVirtualMachine&serviceOfferingId=1&diskOfferingId=1&templateId=2&zoneId=4&apiKey=miVr6X7u6bN\_sdahOBpjNejPgEsT35eXq-jB8CG20YI3yaxXcgpyuaIRmFI\_EJTVwZ0nUkkJbPmY3y2bciKwFQ&signature

=Lxx1DM40AjcXU%2FcaiK8RAP0O1hU%3D

Or in a more readable format:

1. http://localhost:8080/client/api
2. ?command=deployVirtualMachine
3. &serviceOfferingId=1
4. &diskOfferingId=1
5. &templateId=2
6. &zoneId=4
7. &apiKey=miVr6X7u6bN\_sdahOBpjNejPgEsT35eXqjB8CG20YI3yaxXcgpyuaIRmFI\_EJTVwZ0nUkkJbPmY3y2bciKwFQ
8. &signature=Lxx1DM40AjcXU%2FcaiK8RAP0O1hU%3D

The first line is the CloudStack API URL. This is the Cloud instance you wish to interact with.

The second line refers to the command you wish to execute. In our example, we are attempting to deploy a fresh new virtual machine. It is preceded by a (?) to separate itself from the CloudStack API URL.

Lines 3-6 are the required (or optional) parameters for this given command. To see the command and its request parameters, please refer to the appropriate section in the CloudStack API documentation. Each parameter field-value pair (field=value) is separated by a (&).

Line 7 is the user API Key that uniquely identifies the account. See section 2.1 for more information on signing the API request.

Line 8 is the signature hash created to authenticate the user account executing the API command. See section 2.1 for more information on signing the API request.

## Signing API Requests

Whether you access CloudStack API with HTTP or HTTPS, it must still be signed so that CloudStack can verify the call has been authenticated and authorized to execute the command. Make sure that you have both the API Key and Secret Key as provided by the CloudStack administrator for your account before proceeding with the signing process.

To sign a request, we will re-use the above example.

http://localhost:8080/client/api?command=deployVirtualMachine&serviceOfferingId=1&diskOfferingId=1&templateId=2&zoneId=4&apiKey=miVr6X7u6bN\_sdahOBpjNejPgEsT35eXq-jB8CG20YI3yaxXcgpyuaIRmFI\_EJTVwZ0nUkkJbPmY3y2bciKwFQ&signature

=Lxx1DM40AjcXU%2FcaiK8RAP0O1hU%3D

Breaking this down, we have three distinct parts to this URL.

* Base URL: This is the base URL to the CloudStack Management Server.

http://localhost:8080

* API Path: This is the path to the API Servlet that processes the incoming requests.

/client/api?

* Command String: This part of the query string comprises of the command, its parameters, and the API Key that identifies the account.

**NOTE: As with all query string parameters of field-value pairs, the “field” component is case insensitive while all “value” values are case sensitive.**

command=deployVirtualMachine&serviceOfferingId=1&diskOfferingId=1&templateId=2&zoneId=4&apiKey=miVr6X7u6bN\_sdahOBpjNejPgEsT35eXq-jB8CG20YI3yaxXcgpyuaIRmFI\_EJTVwZ0nUkkJbPmY3y2bciKwFQ

* Signature: This is the hashed signature of the Base URL that is generated using a combination of the user’s Secret Key and the HMAC SHA-1 hashing algorithm.

&signature=Lxx1DM40AjcXU%2FcaiK8RAP0O1hU%3D

Every API request has the format: **Base URL+API Path+Command String+Signature**

The process to generate the signature which will eventually authenticate and authorize the user to execute the command is as follows.

1. For each field-value pair (as separated by a ‘&’) in the Command String, URL encode each value so that it can be safely sent via HTTP GET.

**NOTE: Make sure all spaces are encoded as “%20” rather than “+”.**

1. Lower case the entire Command String and sort it alphabetically via the field for each field-value pair. The result of this step would look like the following.

apikey=mivr6x7u6bn\_sdahobpjnejpgest35exq-jb8cg20yi3yaxxcgpyuairmfi\_ejtvwz0nukkjbpmy3y2bcikwfq&command=deployvirtualmachine&diskofferingid=1&serviceofferingid=1&templateid=2&zoneid=4

1. Take the sorted Command String and run it through the HMAC SHA-1 hashing algorithm (most programming languages offer a utility method to do this) with the user’s Secret Key. Base64 encode the resulting byte array in UTF-8 so that it can be safely transmitted via HTTP. The final string produced after Base64 encoding should be “**Lxx1DM40AjcXU%2FcaiK8RAP0O1hU%3D”.**

By reconstructing the final URL using the format of **Base URL+API Path+Command String+Signature**, the final URL should look like:

http://localhost:8080/client/api?command=deployVirtualMachine&serviceOfferingId=1&diskOfferingId=1&templateId=2&zoneId=4&apiKey=miVr6X7u6bN\_sdahOBpjNejPgEsT35eXq-jB8CG20YI3yaxXcgpyuaIRmFI\_EJTVwZ0nUkkJbPmY3y2bciKwFQ&signature

=Lxx1DM40AjcXU%2FcaiK8RAP0O1hU%3D

# Response Formats

CloudStack supports two formats as the response to an API call. The default response is XML. If you would like the response to be in JSON, just add **&response=json** to the Command String.

Sample XML Response:

<listipaddressesresponse>

<allocatedipaddress>

<ipaddress>192.168.10.141</ipaddress>

<allocated>2009-09-18T13:16:10-0700</allocated>

<zoneid>4</zoneid>

<zonename>WC</zonename>

<issourcenat>true</issourcenat>

</allocatedipaddress>

</listipaddressesresponse>

Sample JSON Response:

{ "listipaddressesresponse" :

{ "allocatedipaddress" :

[

{

"ipaddress" : "192.168.10.141",

"allocated" : "2009-09-18T13:16:10-0700",

"zoneid" : "4",

"zonename" : "WC",

"issourcenat" : "true"

}

]

}

}

## Error Handling

If an error occurs while processing an API request, the appropriate response in the format specified is returned. Each error response consists of an error code and an error text describing what possibly can go wrong.

An HTTP error code of 401 is always returned if API request was rejected due to bad signatures, missing API Keys, or the user simply did not have the permissions to execute the command.

# Asynchronous Commands

Starting in CloudStack 2.x, the notion of asynchronous commands has been introduced. Commands are designated as asynchronous when they can potentially take a long period of time to complete such as creating a snapshot or disk volume. They differ from synchronous commands by the following:

* They are identified in the API Reference by an (A).
* They will immediately return a job ID to refer to the job that will be responsible in processing the command.
* If executed as a "create" resource command, it will return the resource ID as well as the job ID.
* You can periodically check the status of the job by making a simple API call to the command, queryAsyncJobResult and passing in the job ID.

## Job Status

The key to using an asynchronous command is the job ID that is returned immediately once the command has been executed. With the job ID, you can periodically check the job status by making calls to queryAsyncJobResult command. The command will return three possible job status integer values:

* 0 - Job is still in progress. Continue to periodically poll for any status changes.
* 1 - Job has successfully completed. The job will return any successful response values associated with command that was originally executed.
* 2 - Job has failed to complete.  Please check the <jobresultcode> tag for failure reason code and <jobresult> for the failure reason.

## Example

The following shows an example of using an asynchronous command.

Assume the API command:

command=deployVirtualMachine&zoneId=1&serviceOfferingId=1&diskOfferingId=1&templateId=1

CloudStack will immediately return a job ID and any other additional data.

<deployvirtualmachineresponse>

  <jobid>1</jobid>

  <id>100</id>

</deployvirtualmachineresponse>

Using, the job ID, you can periodically poll for the results by using the queryAsyncJobResult command.

command=queryAsyncJobResult&jobId=1

Three possible results could come from this query.

**Job is still pending:**

<queryasyncjobresult>

<jobid>1</jobid>

<jobstatus>0</jobstatus>

<jobprocstatus>1</jobprocstatus>

<queryasyncjobresult>

**Job has failed:**

<queryasyncjobresult>

<jobid>1</jobid>

<jobstatus>2</jobstatus>

<jobprocstatus>0</jobprocstatus>

<jobresultcode>551</jobresultcode>

<jobresulttype>text</jobresulttype>

<jobresult>

Unable to deploy virtual machine id = 100 due to not enough capacity

</jobresult>

<queryasyncjobresult>

**Job has succeeded:**

<queryasyncjobresultresponse cloud-stack-version="2.2.1.66">

<jobid>1</jobid>

<jobstatus>1</jobstatus>

<jobprocstatus>0</jobprocstatus>

<jobresultcode>0</jobresultcode>

<jobresulttype>object</jobresulttype>

<jobresult>

<virtualmachine>

<id>450</id>

<name>i-2-450-VM</name>

<displayname>i-2-450-VM</displayname>

<account>admin</account>

<domainid>1</domainid>

<domain>ROOT</domain>

<created>2011-03-10T18:20:25-0800</created>

<state>Running</state>

<haenable>false</haenable>

<zoneid>1</zoneid>

<zonename>San Jose 1</zonename>

<hostid>2</hostid>

<hostname>905-13.sjc.lab.vmops.com</hostname>

<templateid>1</templateid>

<templatename>CentOS 5.3 64bit LAMP</templatename>

<templatedisplaytext>CentOS 5.3 64bit LAMP</templatedisplaytext>

<passwordenabled>false</passwordenabled>

<serviceofferingid>1</serviceofferingid>

<serviceofferingname>Small Instance</serviceofferingname>

<cpunumber>1</cpunumber>

<cpuspeed>500</cpuspeed>

<memory>512</memory>

<guestosid>12</guestosid>

<rootdeviceid>0</rootdeviceid>

<rootdevicetype>NetworkFilesystem</rootdevicetype>

<nic>

<id>561</id>

<networkid>205</networkid>

<netmask>255.255.255.0</netmask>

<gateway>10.1.1.1</gateway>

<ipaddress>10.1.1.225</ipaddress>

<isolationuri>vlan://295</isolationuri>

<broadcasturi>vlan://295</broadcasturi>

<traffictype>Guest</traffictype>

<type>Virtual</type>

<isdefault>true</isdefault>

</nic>

<hypervisor>XenServer</hypervisor>

</virtualmachine>

</jobresult>

</queryasyncjobresultresponse>

# What’s New?

The following describes any new major features of each CloudStack version as it applies to API usage.

## What’s New in 2.2.x?

The biggest change to API usage is the introduction of Basic vs. Advanced Networking in 2.2.x. Our Installation and Admin Guide goes into more detail of how such networks can be configured in CloudStack. The purpose of this section is to explain how the deployment of a Virtual Machine differs depending on which network model CloudStack is setup to use.

Let’s first look at the possible parameters you can pass along with the deployVirtualMachine command.

|  |  |  |
| --- | --- | --- |
| Parameter Name | Description | Required |
| serviceofferingid | **the ID of the service offering for the virtual machine** | **true** |
| templateid | **the ID of the template for the virtual machine** | **true** |
| zoneid | **availability zone for the virtual machine** | **true** |
| *account* | *an optional account for the virtual machine. Must be used with domainId.* | *false* |
| *diskofferingid* | *the ID of the disk offering for the virtual machine. If the template is of ISO format, the diskOfferingId is for the root disk volume. Otherwise this parameter is used to indicate the offering for the data disk volume. If the templateId parameter passed is from a Template object, the diskOfferingId refers to a DATA Disk Volume created. If the templateId parameter passed is from an ISO object, the diskOfferingId refers to a ROOT Disk Volume created.* | *false* |
| *displayname* | *an optional user generated name for the virtual machine* | *false* |
| *domainid* | *an optional domainId for the virtual machine. If the account parameter is used, domainId must also be used.* | *false* |
| *group* | *an optional group for the virtual machine* | *false* |
| *hypervisor* | *the hypervisor on which to deploy the virtual machine* | *false* |
| *name* | *host name for the virtual machine* | *false* |
| *networkids* | *list of network ids used by virtual machine* | *false* |
| *securitygroupids* | *comma separated list of security groups id that going to be applied to the virtual machine. Should be passed only when vm is created from a zone with Basic Network support* | *false* |
| *size* | *the arbitrary size for the DATADISK volume. Mutually exclusive with diskOfferingId* | *false* |
| *userdata* | *an optional binary data that can be sent to the virtual machine upon a successful deployment. This binary data must be base64 encoded before adding it to the request. Currently only HTTP GET is supported. Using HTTP GET (via querystring), you can send up to 2KB of data after base64 encoding.* |  |

Regardless of network configuration, the serviceofferingid, templateid, and zoneid are required parameters on every deployVirtualMachine command call.

CloudStack instance has been configured to use Basic Networking

The only parameter that is possibly required is if your CloudStack instance has been configured to support security groups or not. Please again check with the administrator. If security groups are enabled, the parameter, “securitygroupsids” is required and at least one must be passed in to isolate the newly created Virtual Machine. This can be passed in as a comma delimited list of ids.

CloudStack instance has been configured to use Advanced Networking

The parameter, “networkids”, is a required parameter and IDs can be passed in as a comma delimited list.

The logic to determine what appropriate network ID(s) to deploy your virtual machine on is as follows (all examples are using sample values. Please make sure you use the correct IDS as returned by the API):

1. First list all available networks that are available to the account you wish to create the virtual machine for. Two calls are currently required to find all available networks.

Find all networks dedicated to the account:

command=listNetworks&domainid=1&account=admin&zoneid=1

Find all networks are that shared (public) to the account:

command=listNetworks&isshared=true&zoneid=1

1. If there are no networks returned, then you must be prepared to create the virtual guest network for the account. (In 2.2.3 and beyond, this will no longer be required and automatically created). To create this network, do the following.
   1. Execute listNetworkOffering to find the network offering for the virtual guest network.

command=listNetworkOfferings&guestiptype=Virtual

* 1. This should return you one network offering. Make sure the “availability” response value of the network offering is set to “required”. Use the network offering ID to create a virtual guest network.

command=createNetwork&networkofferingid=1&name=networkname&displaytext=networkdescription&zoneid=1

* 1. The response to the createNetwork command will return you a network object and you can then use this network ID for your deployVirtualMachine command.

1. If there are one or more network returned, iterate through all the networks and you must pass at least one default network where the response attribute, “isdefault” is set to “true” to the deployVirtualMachine command. If there are no default networks, you must use **step 2** to create a virtual guest network.

The final command for a deployVirtualMachine command using advanced networking could look like:

command=deployVirtualMachine&zoneId=1&templateid=204&serviceofferingid=1&networkids=205

**NOTE: In 2.2.3 and beyond, this process will be simplified such that networks are automatically created if none are passed in.**