2.2 CloudStack API Developer Guide

Document revised February 17, 2012

© 2011, 2012 Citrix Systems, Inc. All rights reserved. Specifications are subject to change without notice. Citrix Systems, Inc., the Citrix logo, Citrix XenServer, Citrix XenCenter, and CloudStack are trademarks or registered trademarks of Citrix Systems, Inc. All other brands or products are trademarks or registered trademarks of their respective holders.

CloudStack software includes code redistributed under Apache Software Foundation license <http://www.apache.org/licenses/LICENSE-2.0>.

Contents

[1 Introduction 4](#_Toc317214163)

[1.1 Getting Started 4](#_Toc317214164)

[2 Making API Requests 5](#_Toc317214165)

[2.1 Signing API Requests 5](#_Toc317214166)

[3 Responses 7](#_Toc317214167)

[3.1 Response Formats: XML and JSON 7](#_Toc317214168)

[3.2 Maximum Result Pages Returned 7](#_Toc317214169)

[3.3 Error Handling 8](#_Toc317214170)

[4 Asynchronous Commands 9](#_Toc317214171)

[4.1 Job Status 9](#_Toc317214172)

[4.2 Example 9](#_Toc317214173)

[5 Working With Usage Data 12](#_Toc317214174)

[5.1 Usage Record Format 12](#_Toc317214175)

[5.2 Usage Types 14](#_Toc317214176)

[5.3 Example response from listUsageRecords 15](#_Toc317214177)

[5.4 Dates in the Usage Record 15](#_Toc317214178)

[6 Alerts 18](#_Toc317214179)

[7 Time Zones 19](#_Toc317214180)

[8 What’s New? 20](#_Toc317214181)

[8.1 What’s New in 2.2.x? 20](#_Toc317214182)

[*8.1.1* *Basic Networking* 20](#_Toc317214183)

[*8.1.2* *Advanced Networking* 21](#_Toc317214184)

# Introduction

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

* **Root 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>

## Getting Started

To get started using the CloudStack API, you should have the following:

* URL of 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.
* Familiarity with HTTP GET/POST and query strings.
* Knowledge of either XML or JSON.
* Knowledge of a programming language that can generate HTTP requests; for example, Java or PHP.

# 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 (for example, <http://www.cloud.com:8080/client/api>)
* Command: The web services command you wish to execute, such as start a virtual machine or 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 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 preceded by an ampersand character (&).

Line 7 is the user API Key that uniquely identifies the account. See Signing API Requests on page 5.

Line 8 is the signature hash created to authenticate the user account executing the API command. See Signing API Requests on page 5.

## Signing API Requests

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

To show how to sign a request, we will re-use the previous 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 several 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.

To generate the signature.

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 in the format Base URL+API Path+Command String+SignaEFASSture, 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

# Responses

This section describes what to expect with the responses to your API requests.

## Response Formats: XML and JSON

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, 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"

}

]

}

}

## Maximum Result Pages Returned

For each cloud, there is a default upper limit on the number of results that any API command will return in a single page. This is to help prevent overloading the cloud servers and prevent DOS attacks. For example, if the page size limit is 500 and a command returns 10,000 results, the command will return 20 pages.

The default page size limit can be different for each cloud. It is set in the global configuration parameter default.page.size. If your cloud has many users with lots of VMs, you might need to increase the value of this parameter. At the same time, be careful not to set it so high that your site can be taken down by an enormous return from an API call. For more information about how to set global configuration parameters, see "Describe Your Deployment" in the Installation Guide.

To decrease the page size limit for an individual API command, override the global setting with the page and pagesize parameters, which are available in any list\* command (listCapabilities, listDiskOfferings, etc.).

* Both parameters must be specified together.
* The value of the pagesize parameter must be smaller than the value of default.page.size. That is, you can not increase the number of possible items in a result page, only decrease it.

For syntax information on the list\* commands, see the API Reference.

## 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. For an example error response, see page 10.

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 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>

**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>

# Working With Usage Data

The Usage Server provides aggregated usage records which you can use to create billing integration for the CloudStack platform. The Usage Server works by taking data from the events log and creating summary usage records that you can access using the listUsageRecords API call.

The usage records show the amount of resources, such as VM run time or template storage space, consumed by guest instances. In the special case of bare metal instances, no template storage resources are consumed, but records showing zero usage are still included in the Usage Server's output.

The Usage Server runs at least once per day. It can be configured to run multiple times per day. Its behavior is controlled by configuration settings as described in the CloudStack Administration Guide.

## Usage Record Format

For running and allocated virtual machine usage, and for network usage (bytes sent/received), the following fields exist in a usage record.

* **account** – name of the account
* **accountid** – ID of the account
* **domainid** – ID of the domain in which this account resides
* **zoneid** – Zone where the usage occurred
* **description** – A string describing what the usage record is tracking
* **usage** – String representation of the usage, including the units of usage (e.g. 'Hrs' for VM running time)
* **usagetype** – A number representing the usage type (see Usage Types on page 14)
* **rawusage** – A number representing the actual usage; for types 1 and 2 the usage will be hours, and for types 4 and 5 the usage will be bytes
* **virtualMachineId** – For usage types 1 and 2, the ID of the virtual machine
* **name** – For usage types 1 and 2, the name of the virtual machine is included
* **offeringid** – For usage types 1 and 2, the ID of the service offering
* **templateid** – For usage types 1 and 2, the ID of the template or (v2.2.8 and greater) the ID of the parent template. The parent template value is present when the current template was created from a volume.
* **usageid** – ID of the usage record
* **type** – Hypervisor
* **startdate, enddate** – The range of time for which the usage is valid; see Dates in the Usage Record on page 15

For IP address usage the following fields exist in a usage record.

* **account** – name of the account
* **accountid** – ID of the account
* **domainid** – the id of the domain in which this account resides
* **zoneid** – the zone where the usage occurred
* **description** – a string describing what the usage record is tracking
* **usagetype** – a number representing the usage type (see Usage Types on page 14)
* **usageid** – ID of the usage record
* **ipaddress** – the ip address owned by the account
* **assigndate** – the date the address was assigned to the account
* **releasedate** – the date the address was released from the account (if the address is still owned by the account, this won't be present)
* **issourcenat** – Whether source NAT is enabled for the IP address

For disk volumes, templates, ISOs, and snapsnots, the following fields exist in a usage record.

* **account** – name of the account
* **accountid** – ID of the account
* **domainid** – ID of the domain in which this account resides
* **zoneid** – Zone where the usage occurred
* **description** – A string describing what the usage record is tracking
* **usage** – String representation of the usage, including the units of usage (e.g. 'Hrs' for VM running time)
* **usagetype** – A number representing the usage type (see Usage Types on page 14)
* **rawusage** – A number representing the actual usage; for types 1 and 2 the usage will be hours, and for types 4 and 5 the usage will be bytes
* **offeringid** – For usage types 1 and 2, the ID of the service offering
* **usageid** – ID of the usage record
* **size** – The amount of storage allocated
* **startdate, enddate** – The range of time for which the usage is valid; see Dates in the Usage Record on page 15

## Usage Types

The following table shows all usage types in the CloudStack platform.

|  |  |  |
| --- | --- | --- |
| **Type ID** | **Type Name** | **Description** |
| **1** | RUNNING\_VM | Tracks the total running time of a VM per usage record period.  If the VM is upgraded during the usage period, you will get a separate Usage Record for the new upgraded VM. |
| **2** | ALLOCATED\_VM | Tracks the total time the VM has been created to the time when it has been destroyed.  This usage type is also useful in determining usage for specific templates such as Windows-based templates. |
| **3** | IP\_ADDRESS | Tracks the public IP address owned by the account. |
| **4** | NETWORK\_BYTES\_SENT | Tracks the total number of bytes sent by all the VMs for an account.  Cloud.com does not currently track network traffic per VM. |
| **5** | NETWORK\_BYTES\_RECEIVED | Tracks the total number of bytes received by all the VMs for an account.  Cloud.com does not currently track network traffic per VM. |
| **6** | VOLUME | Tracks the total time a disk volume has been created to the time when it has been destroyed. |
| **7** | TEMPLATE | Tracks the total time a template (either created from a snapshot or uploaded to the cloud) has been created to the time it has been destroyed.  The size of the template is also returned. |
| **8** | ISO | Tracks the total time an ISO has been uploaded to the time it has been removed from the cloud.  The size of the ISO is also returned. |
| **9** | SNAPSHOT | Tracks the total time a snapshot has been created to the time it has been destroyed. |
| **11** | LOAD\_BALANCER\_POLICY | Tracks the total time a load balancer policy has been created to the time it has been removed.  Cloud.com does not track whether a VM has been assigned to a policy. |
| **12** | PORT\_FORWARDING\_RULE | Tracks the time from when a port forwarding rule was created until the time it was removed. |
| **13** | NETWORK\_OFFERING | The time from when a network offering was assigned to a VM until it is removed. |
| **14** | VPN\_USERS | The time from when a VPN user is created until it is removed. |

## Example response from listUsageRecords

<listusagerecordsresponse>

<count>1816</count>

<usagerecord>

<account>user5</account>

<accountid>10004</accountid>

<domainid>1</domainid>

<zoneid>1</zoneid>

<description>

i-3-4-WC running time (ServiceOffering: 1) (Template: 3)

</description>

<usage>2.95288 Hrs</usage>

<usagetype>1</usagetype>

<rawusage>2.95288</rawusage>

<virtualmachineid>4</virtualmachineid>

<name>i-3-4-WC</name>

<offeringid>1</offeringid>

<templateid>3</templateid>

<usageid>245554</usageid>

<type>XenServer</type>

<startdate>2009-09-15T00:00:00-0700</startdate>

<enddate>2009-09-18T16:14:26-0700</enddate>

</usagerecord>

… (1,815 more usage records)

</listusagerecordsresponse>

## Dates in the Usage Record

IP address usage records will have an assigned date and released date. The assigned date is when the address was assigned to the users account, and the released date is when the address was released from the users account. If the released date is not present, the IP address is currently held by the user. If a call to listUsageRecords is made, any IP address held during that range will be returned. Here are some examples:

Example 1: IP address 192.168.0.4 is assigned to account user5 on October 15th, 2009 and is currently held by the account.

listUsageRecords&startDate=2009-10-01&endDate=2009-10-31

<usagerecord>

<account>user5</account>

<accountid>10004</accountid>

<domainid>1</domainid>

<zoneid>1</zoneid>

<description>network bytes received</description>

<usagetype>5</usagetype>

<ipaddress>192.168.0.4</ipaddress>

<assignedDate>2009-10-15T12:31:15-0700</assignedDate>

</usagerecord>

Since the assigned date is after October 1st, 2009, you will need to compute usage (in minutes, hours, however you'd like) from 2009-10-15T12:31:15-0700 to 2009-10-31T23:59:59-0700.

listUsageRecords&startDate=2009-10-14&endDate=2009-10-16

<usagerecord>

<account>user5</account>

<accountid>10004</accountid>

<domainid>1</domainid>

<zoneid>1</zoneid>

<description>network bytes received</description>

<usagetype>5</usagetype>

<ipaddress>192.168.0.4</ipaddress>

<assignedDate>2009-10-15T12:31:15-0700</assignedDate>

</usagerecord>

Since the assigned date is after October 14th, 2009, you will need to compute usage (in minutes, hours, however you'd like) from 2009-10-15T12:31:15-0700 to 2009-10-16T23:59:59-0700.

listUsageRecords&startDate=2009-11-01&endDate=2009-11-30

<usagerecord>

<account>user5</account>

<accountid>10004</accountid>

<domainid>1</domainid>

<zoneid>1</zoneid>

<description>network bytes received</description>

<usagetype>5</usagetype>

<ipaddress>192.168.0.4</ipaddress>

<assignedDate>2009-10-15T12:31:15-0700</assignedDate>

</usagerecord>

Since the assigned date is before November 1st, 2009, you will need to compute usage (in minutes, hours, however you'd like) from 2009-11-01T00:00:00-0700 to 2009-11-30T23:59:59-0700.

Example 2: IP address 192.168.0.4 is assigned to account user5 on October 15th, 2009 and released on October 18th, 2009.

listUsageRecords&startDate=2009-10-01&endDate=2009-10-31

<usagerecord>

<account>user5</account>

<accountid>10004</accountid>

<domainid>1</domainid>

<zoneid>1</zoneid>

<description>network bytes received</description>

<usagetype>5</usagetype>

<ipaddress>192.168.0.4</ipaddress>

<assignedDate>2009-10-15T12:31:15-0700</assignedDate>

<releasedDate>2009-10-18T18:04:32-0700</releasedDate>

</usagerecord>

Since the assigned date and released dates are in the query range, you will need to compute usage (in minutes, hours, however you'd like) from 2009-10-15T12:31:15-0700 to 2009-10-18T18:04:32-0700.

listUsageRecords&startDate=2009-10-14&endDate=2009-10-16

<usagerecord>

<account>user5</account>

<accountid>10004</accountid>

<domainid>1</domainid>

<zoneid>1</zoneid>

<description>network bytes received</description>

<usagetype>5</usagetype>

<ipaddress>192.168.0.4</ipaddress>

<assignedDate>2009-10-15T12:31:15-0700</assignedDate>

<releasedDate>2009-10-18T18:04:32-0700</releasedDate>

</usagerecord>

Since the released date is after October 16th, 2009, you will need to compute usage (in minutes, hours, however you'd like) from 2009-10-15T12:31:15-0700 to 2009-10-16T23:59:59-0700.

listUsageRecords&startDate=2009-11-01&endDate=2009-11-30

- no results since the released date is before November 1st, 2009

Non-IP address records will have a start date and end date. These dates define the range in which the raw usage number was calculated. If daily aggregation is used, the start date will typically be midnight on the day in question and end date will be 23:59:59 on the day in question. A virtual machine could have been deployed at noon on that day, stopped at 6pm on that day, then started up again at 11pm. When usage is calculated on that day there will be 7 hours of a running VM (usage type 1). For an allocated VM (usage type 2) there will be 12 hours of usage. If the same virtual machine runs for the entire next day, there will 24 hours of both running VM (usage type 1) and allocated VM (usage type 2). Note: the start date is not the time a virtual machine was started, nor is the end date the time when a virtual machine was stopped. They merely correspond to the time range when usage was calculated.

For network usage, the start date and end date again define the range in which the bytes transferred were calculated. If a user downloads 10 MB and uploads 1 MB in one day, then there will be two records, one showing the 10 megabytes received and one showing the 1 megabyte sent.

There is an exception to when the start date and/or end date do not correspond to midnight and 11:59:59pm respectively when daily aggregation is used. This occurs when the usage server has more than one day's worth of unprocessed data. Since the start/end dates show the time in which the usage was calculated, the old data will be pulled in and the aggregation time will show the start of the earliest event (VM start, bytes sent, etc.).

# Alerts

The following is the list of alert type numbers. These can be returned by listAlerts.

MEMORY = 0

CPU = 1

STORAGE =2

STORAGE\_ALLOCATED = 3

PUBLIC\_IP = 4

PRIVATE\_IP = 5

HOST = 6

USERVM = 7

DOMAIN\_ROUTER = 8

CONSOLE\_PROXY = 9

ROUTING = 10// lost connection to default route (to the gateway)

STORAGE\_MISC = 11 // lost connection to default route (to the gateway)

USAGE\_SERVER = 12 // lost connection to default route (to the gateway)

MANAGMENT\_NODE = 13 // lost connection to default route (to the gateway)

DOMAIN\_ROUTER\_MIGRATE = 14

CONSOLE\_PROXY\_MIGRATE = 15

USERVM\_MIGRATE = 16

VLAN = 17

SSVM = 18

USAGE\_SERVER\_RESULT = 19

# Time Zones

The following time zone identifiers are accepted by the CloudStack API. There are several places that have a time zone as a required or optional parameter. These include scheduling recurring snapshots, creating a user, and specifying the usage time zone in the Configuration table.

Etc/GMT+12

Etc/GMT+11

Pacific/Samoa

Pacific/Honolulu

US/Alaska

America/Los\_Angeles

Mexico/BajaNorte

US/Arizona

US/Mountain

America/Chihuahua

America/Chicago

America/Costa\_Rica

America/Mexico\_City

Canada/Saskatchewan

America/Bogota

America/New\_York

America/Caracas

America/Asuncion

America/Cuiaba

America/HalifaxAmerica/La\_Paz

America/Santiago

America/St\_Johns

America/Araguaina

America/Argentina/Buenos\_Aires

America/Cayenne

America/Godthab

America/Montevideo

Etc/GMT+2

Atlantic/Azores

Atlantic/Cape\_Verde

Africa/Casablanca

Etc/UTC

Atlantic/Reykjavik

Europe/London

CET

Europe/Bucharest

Africa/Johannesburg

Asia/Beirut

Africa/CairoAsia/Jerusalem

Europe/Minsk

Europe/Moscow

Africa/Nairobi

Asia/Karachi

Asia/Kolkata

Asia/Bangkok

Asia/Shanghai

Asia/Kuala\_Lumpur

Australia/Perth

Asia/Taipei

Asia/Tokyo

Asia/Seoul

Australia/Adelaide

Australia/Darwin

Australia/Brisbane

Australia/Canberra

Pacific/Guam

Pacific/Auckland

# 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 Guide and Administrator's Guide go 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 set up 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.

### Basic Networking

When the 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.

### Advanced Networking

When the 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.**