Rhea Cloud v0.99

Thank you for downloading Rhea Cloud, a ColdFusion Package developed to interact with the Rackspace Cloud Server and Cloud Files APIs.

For information on the latest releases, please visit http://www.fuzzyorange.co.uk

License and Credits

Copyright 2010 Fuzzy Orange Ltd (http://www.fuzzyorange.co.uk)

Licensed under the Apache License, Version 2.0 (the "License");
You may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.

For full License details, please read the LICENSE file available with this download.

Authors

Developed by Matt Gifford AKA coldfumonkeh
Senior Developer, Fuzzy Orange Ltd
http://www.fuzzyorange.co.uk
http://www.mattgifford.co.uk

Requirements

Rhea Cloud requires ColdFusion MX7

Installation

Unzip the package archive to the desired location (typically in the web root). After installation, you will see the following directories:

- Installation (contains this installation guide, no more, no less)
- com (contains all of the ColdFusion Components required for the Rhea Application to interact with the Rackspace API)

The application does not interact with any database, and as such needs no datasources.

Getting Started

To interact with the Rackspace Cloud API, you should ideally have an account with Rackspace for either the Cloud Server, Cloud Files package or both.
Visit http://www.rackspacecloud.com/ for more information on these products.

Once you have an account, you need to obtain your Cloud Key, typically a 33 character UUID API Key, which will let you access the API.

Open up Application.cfc, and within the onApplicationStart() method edit the code to include your Cloud Username and API key, as provided by Rackspace:

```
<cffunction name="onApplicationStart" output="false">
	<cfscript>
		this.cloudUser	= '<your username here>';
		this.cloudKey	= '<your api key here>';

	Application.objRackspace = createObject('component',
		'com.fuzzyorange.rackspaceCloud').
		init(username=this.cloudUser,apiKey=this.cloudKey);
	</cfscript>
</cffunction>
```

The Rackspace object is then instantiated and persisted in the Application scope, sending through the account details you have supplied.

The rackspaceCloud.cfc file is the main object, and acts as a façade or Service Layer to interact with the underlying components.
Upon instantiation, an authentication method is immediately run using the account details supplied to verify against the Cloud API. An unsuccessful response will display an error message and abort any further processing:

```
"Authentication failed. Please check the User and API Key details are correct"
```

Response Formats

The rackspaceCloud.cfc accepts a third non-required parameter called ‘format’, which defines the return format of responses from the API. The options available are ‘XML’ (default) or ‘JSON’.

To receive responses in JSON for every method, simply add the third parameter into the instantiation call:

```
Application.objRackspace = createObject('component',
      'com.fuzzyorange.rackspaceCloud').
      init(username=this.cloudUser,apiKey=this.cloudKey,format='JSON'
);
```

As mentioned, the global response format is available from the moment on instantiation and is applicable to every method called from the component.

Overwriting the format

Every public-facing method will use the global return format, but each method also allows the user to specify an individual response format should you wish to overwrite the global option for any specific function.

To do so, simply add the ‘format’ parameter to any method call and specify either ‘XML’ or ‘JSON’ as the value.

Separating the API components

Rhea Cloud is separated into two ‘interfaces’;

1. the Rackspace Cloud Server API
2. the Rackspace Cloud Files API

Although not typically an Interface Object in terms of Object-Oriented Design, the rackspaceCloud.cfc façade loads and stores these components during instantiation, and both contain the methods relevant to the particular API they deal with.

Accessing the ‘Interface’ Objects

Access to the interface objects is made through the rackspaceCloud.cfc.

In the below example, the code has been added to the onApplicationStart() method in Application.cfc to persist the objFiles and objServer objects within the Application scope.

```
<cffunction name="onApplicationStart" output="false">
	<cfscript>
		this.cloudUser	= 'gorillaworks';
		this.cloudKey	= '928220021f98822e918efcdb8fb33a40';

		Application.objRackspace = createObject('component',
			'com.fuzzyorange.rackspaceCloud').
			init(username=this.cloudUser,apiKey=this.cloudKey);

		// persist the specific 'interface'
		// objects in the Application scope

		Application.objFiles	=
			Application.objRackspace.fileInterface();

		Application.objServer	=
			Application.objRackspace.serverInterface();
	</cfscript>
</cffunction>
```

It is through these two objects that the public-facing methods for each API are available.

Examples

When creating a Cloud server with Rackspace, you have the option to choose from a wide selection of 'flavors' and 'images' to create the server that best suits your needs and requirements; the image reflects the choice of Operating System you wish to implement, and the flavor covers the choice of HDD space and RAM.

List Flavors

Let's run a call to the server API to pull out the list of flavors available.

```
<cfset rheaFlavors = Application.objServer.listFlavors() />

<cfdump var="#rheaFlavors#" />
```

The returned object from the call, as with the majority of the method responses, will be in the format of a structure containing the data, the response message and a Boolean success value, as seen in the example response below:

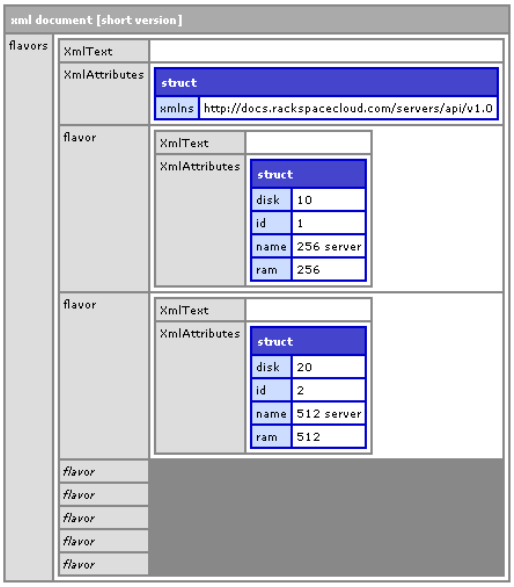
struct	
DATA	<?xml version="1.0" encoding="UTF-8" standalone="yes"?><flavors xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"><flavor disk= server" id="4"/><flavor disk="160" ram="4096" name="4GB server" id="5"/><flavor disk="320" ram="8192" name="8GB server" id="6"/><
MESSAGE	200 OK
SUCCESS	true

To access the data returned from the method call, simply access the ‘data’ struct key:

```
<cfset rheaFlavors = Application.objServer.listFlavors().data />

<cfdump var="#XmlParse(rheaFlavors)#" />
```

An example XmlParse() of the returned rheaFlavors response is shown below:

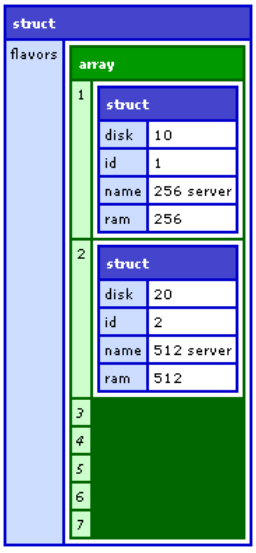


Overwriting the return format in the same method call to obtain a JSON response:

```
<cfset rheaFlavors =
Application.objServer.listFlavors(format='json').data />

<cfdump var="#DeserializeJSON(rheaFlavors)#" />
```

An example DeserializeJSON () of the returned rheaFlavors response is shown below:

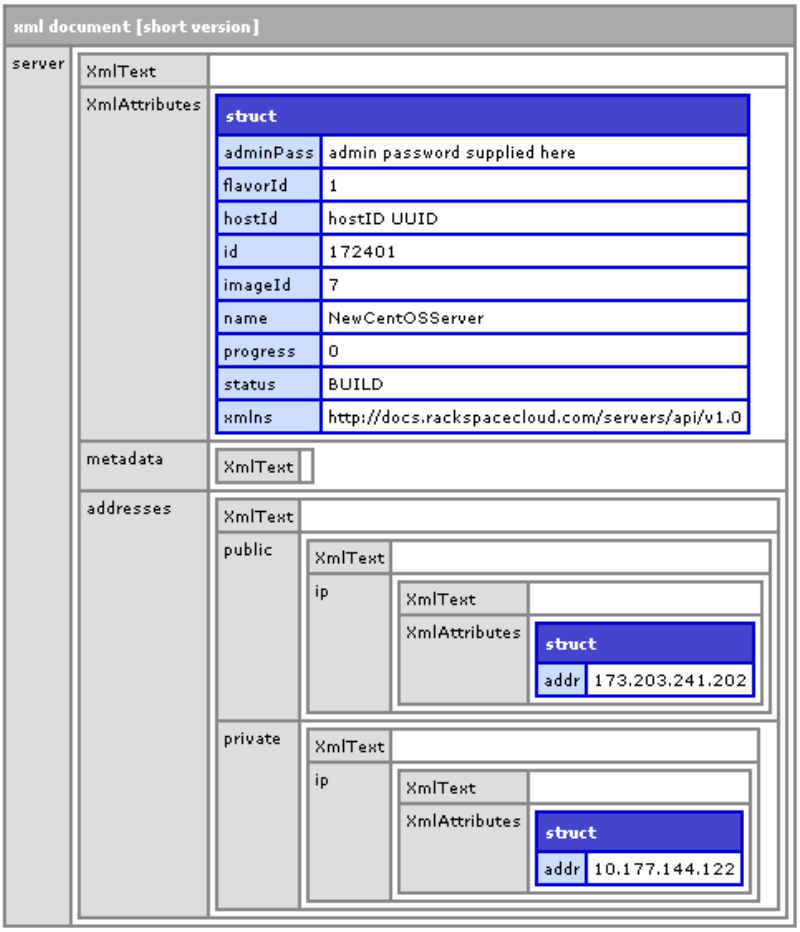


Creating a server

Below is an example of creating a server using the API.

```
<cfset createServer = Application.objServer.createServer(
                        name='New CentOS Server',
                        flavorID='1',
                        imageID='7') />
```

Simply passing through the name of the server, the flavorID and the imageID, Rhea Cloud transmits the data to the Rackspace API and returns a response similar to the one shown below:



Available File Methods

Here is a list of all available methods within the Rhea Cloud package to interact with the Rackspace Cloud Files API.

These methods are available through the file interface like so:

```
Application.objFiles.methodname(param1,param2..)
```

createContainer

I create a Container.

Parameters:
containerName: string, required, containerName - Name of the container you wish to create
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

deleteContainer

This method permanently removes a Container

Parameters:
containerName: string, required, containerName - Name of the container you wish to delete
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

deleteObject

I permanently remove the specified Object from the storage system (metadata and data)

Parameters:
containerName: string, required, containerName - Name of the Container that contains the Object you wish to retrieve.
objectName: string, required, objectName - The name of the Object
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

getAllContainerDetails

I determine the number of Containers within the account and the total bytes stored.

Parameters:
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

getCDNContainerAttributes
I return the CDN attributes of the supplied CDN-enabled Container.

Parameters:
containerName: string, required, containerName - Name of the container you wish to retrieve data for.
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

getCDNContainers
I return a list of CDN-enabled Containers.

Parameters:
limit: string, optional, limit - For an integer value N, limits the number of results to at most N values.
marker: string, optional, marker - Given a string value X, return Object names greater in value than the specified marker.
enabled_only: boolean, optional, enabled_only - Set to 'true' to return only the CDN-enabled Containers
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

getContainerDetails
I determine the number of Objects and total stored bytes within the Container

Parameters:
containerName: string, required, containerName - Name of the container you wish to create
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

getContainers
I return a list of storage Containers applied to the account.

Parameters:
limit: string, optional, limit - For an integer value N, limits the number of results to at most N values.
marker: string, optional, marker - Given a string value X, return Object names greater in value than the specified marker.
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

getObjectMeta
I retrieve an Object's metadata

Parameters:
containerName: string, required, containerName - Name of the Container that contains the Object you wish to retrieve.
objectName: string, required, objectName - The name of the Object
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

getObjectsInContainer
I retrieve a list of Objects stored in the selected Container

Parameters:
containerName: string, required, containerName - Name of the container you wish to retrieve listings for.
limit: string, optional, limit - For an integer value N, limits the number of results to at most N values.
marker: string, optional, marker - Given a string value X, return Object names greater in value than the specified marker.
prefix: string, optional, prefix - For a string value X, causes the results to be limited to Object names beginning with the substring X.
path: string, optional, path - For a string value X, return the Object names nested in the pseudo path (assuming preconditions are met)
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

putObject
I am used to write, or overwrite, an Object's metadata and content

Parameters:
containerName: string, required, containerName - Name of the container you wish to place the object into.
object: Any, required, object - The Object data
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

setCDNContainer
I CDN-enable a Container and set it's attributes.

Parameters:
containerName: string, required, containerName - Name of the container you wish to enable
format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

setCDNContainerAttributes

I am used to adjust the CDN attributes of the supplied CDN-enabled Container.

Parameters:

containerName: string, required, containerName - Name of the container you wish to edit the attributes of.

format: string, optional, format - Specify either JSON or XML to return the respective serialized response.

Available Server Methods

Here is a list of all available methods within the Rhea Cloud package to interact with the Rackspace Cloud Sevrer API.

These methods are available through the server interface like so:

```
Application.objServer.methodname(param1,param2..)
```

confirmResize

During a resize operation, the original server is saved for a period of time to allow roll back if there is a problem. Once the newly resized server is tested and has been confirmed to be functioning properly, use this operation to confirm the resize. After confirmation, the original server is removed and cannot be rolled back to. All resizes are automatically confirmed after 24 hours if they are not explicitly confirmed or reverted.

Parameters:
serverID: string, required, serverID - I am the ID of a specific server you wish to confirm.
format: string, optional, format - The return format of the response. XML or JSON.

createImage

This operation creates a new image for the given server ID. Once complete, a new image will be available that can be used to rebuild or create servers.

Parameters:
serverID: string, required, serverID - I am the ID of a specific server you wish to create an image of.
imageName: string, required, imageName - The name to provide for the created image.
format: string, optional, format - The return format of the response. XML or JSON.

createServer

This operation asynchronously provisions a new server. The progress of this operation depends on several factors including location of the requested image, network i/o, host load, and the selected flavor.

Parameters:
name: string, required, name - I am the name for the new server
flavorID: string, required, flavorID - I am the ID of a specific flavor you wish to use to create the server.
imageID: string, required, imageID - I am the ID of a specific image you wish to use to create the server.
format: string, optional, format - The return format of responses from the cloud

server API. XML or JSON.

createSharedIPGroup

This operation creates a new shared IP group. Please note, all responses to requests for shared_ip_groups return an array of servers. However, on a create request, the shared IP group can be created empty or can be initially populated with a single server. Submitting a create request with a sharedIpGroup that contains an array of servers will generate a badRequest (400) fault.

Parameters:

ipgroupName: string, required, ipgroupName - The name to apply to the new IP group.

serverID: string, required, serverID - I am the ID of a specific server you wish to obtain details for.

format: string, optional, format - The return format of the response. XML or JSON.

createUpdateSchedule

This operation creates a new backup schedule or updates an existing backup schedule for the specified server. Backup schedules will occur only when the enabled attribute is set to true. The weekly and daily attributes can be used to set or to disable individual backup schedules.

Parameters:

serverID: String, required, serverID - I am the ID of a specific server you wish to obtain details for.

enabled: boolean, optional, enabled - Boolean value to set if the backup schedule is enabled.

weekly: String, required, weekly - The weekly backup schedule value; eg THURSDAY

daily: String, required, daily - The daily backup schedule value; eg H_0200_0400

format: string, optional, format - The return format of the response. XML or JSON.

deleteImage

This operation deletes an image from the system.

Parameters:

imageID: string, required, imageID - I am the ID of a specific image you wish to delete.

format: string, optional, format - The return format of the response. XML or JSON.

deleteServer
This operation deletes a cloud server instance from the system.

Parameters:
serverID: string, required, serverID - I am the ID of a specific server you wish to delete.
format: string, optional, format - The return format of the response. XML or JSON.

deleteSharedIPGroup
This operation deletes the specified shared IP group. This operation will ONLY succeed if 1) there are no active servers in the group (i.e. they have all been terminated) or 2) no servers in the group are actively sharing IPs.

Parameters:
groupID: string, required, groupID - The ID of the specific IP group.
format: string, optional, format - The return format of the response. XML or JSON.

disableSchedule
This operation disables the backup schedule for the specified server.

Parameters:
serverID: string, required, serverID - I am the ID of a specific server you wish to obtain details for.
format: string, optional, format - The return format of the response. XML or JSON.

getFlavorDetails
This operation returns details of the specified flavor.

Parameters:
flavorID: string, required, flavorID - I am the ID of a specific flavor you wish to obtain details for.
format: string, optional, format - The return format of the response. XML or JSON.

getImageDetails
This operation returns details of the specified image.

Parameters:
imageID: string, required, imageID - I am the ID of a specific image you wish to obtain details for.
format: string, optional, format - The return format of the response. XML or JSON.

getServerDetails
I will return the details of a specific server.

Parameters:
serverID: string, required, serverID - I am the ID of a specific server you wish to obtain details for.
format: string, optional, format - The return format of the response. XML or JSON.

getSharedIPGroupDetails
This operation returns details of the specified shared IP group.

Parameters:
groupID: string, required, groupID - The ID of the specific IP group.
format: string, optional, format - The return format of the response. XML or JSON.

listFlavors
This operation will list all available flavors with details.

Parameters:
showDetail: boolean, optional, showDetail - If TRUE, will return all details for the flavors, not just IDs and names
format: string, optional, format - The return format of the response. XML or JSON.

listImages
This operation will list all images visible by the account.

Parameters:
showDetail: boolean, optional, showDetail - If TRUE, will return all details for the images, not just IDs and names
format: string, optional, format - The return format of the response. XML or JSON.

listSchedules
I will return a list of the backup schedules for the specified server

Parameters:
serverID: string, required, serverID - I am the ID of a specific server you wish to obtain details for.
format: string, optional, format - The return format of the response. XML or JSON.

listServerAddresses
I will return details of all server addresses for a specific server.

Parameters:
serverID: string, required, serverID - I am the ID of a specific server you wish to obtain details for.
filterList: string, optional, filterList - If set to ALL by default, will return all server addresses. Other options are PUBLIC or PRIVATE, which will return only details for the public or private addresses.
format: string, optional, format - The return format of the response. XML or JSON.

listServers
I will return details of all servers currently associated with the authenticated account.

Parameters:
showDetail: boolean, optional, showDetail - If TRUE, will return all details for the servers, not just IDs and names
format: string, optional, format - The return format of the response. XML or JSON.

listSharedIPGroups
This operation provides a list of shared IP groups associated with your account.

Parameters:
showDetail: boolean, optional, showDetail - If TRUE, will return all details for the IP groups, not just IDs and names
format: string, optional, format - The return format of the response. XML or JSON.

rebootServer
The reboot function allows for either a soft or hard reboot of a server. With a soft reboot (SOFT), the operating system is signaled to restart, which allows for a graceful shutdown of all processes. A hard reboot (HARD) is the equivalent of power cycling the server.

Parameters:
serverID: string, required, serverID - I am the ID of a specific server you wish to reboot.
rebootType: string, optional, rebootType - The type of reboot to perform on the server. SOFT or HARD.
format: string, optional, format - The return format of the response. XML or JSON.

rebuildServer

The rebuild function removes all data on the server and replaces it with the specified image. serverId and IP addresses will remain the same.

Parameters:

serverID: string, required, serverID - I am the ID of a specific server you wish to rebuild.

imageID: string, required, imageID - I am the ID of a specific image you wish to use to rebuild the server.

format: string, optional, format - The return format of the response. XML or JSON.

resizeServer

The resize function converts an existing server to a different flavor, in essence, scaling the server up or down. The original server is saved for a period of time to allow rollback if there is a problem. All resizes should be tested and explicitly confirmed, at which time the original server is removed. All resizes are automatically confirmed after 24 hours if they are not explicitly confirmed or reverted.

Parameters:

serverID: string, required, serverID - I am the ID of a specific server you wish to resize.

flavorID: string, required, flavorID - I am the ID of a specific flavour you wish to use.

format: string, optional, format - The return format of the response. XML or JSON.

revertResize

During a resize operation, the original server is saved for a period of time to allow for roll back if there is a problem. If you determine there is a problem with a newly resized server, use this operation to revert the resize and roll back to the original server. All resizes are automatically confirmed after 24 hours if they have not already been confirmed explicitly or reverted.

Parameters:

serverID: string, required, serverID - I am the ID of a specific server you wish to confirm.

format: string, optional, format - The return format of the response. XML or JSON.

updateServerNamePassword

This operation allows you to update the name of the server and/or change the administrative password. This operation changes the name of the server in the Cloud Servers system and does not change the server host name itself.

Parameters:
serverID: string, required, serverID - I am the ID of a specific server you wish to obtain details for.
serverName: string, optional, serverName - I am the new name for the server image.
adminPassword: string, optional, adminPassword - I am the new admin password.
format: string, optional, format - The return format of the response. XML or JSON.

Limitations

Remaining functionality

shareIPAddress and unshareIPAddress are two methods remaining to be added into the Rhea Cloud package. These two methods deal specifically with the Server API, and will be included in the next release.

Testing

Rhea Cloud has been tested on the Adobe ColdFusion platform.
We have yet to test fully on Railo or OpenBlueDragon.