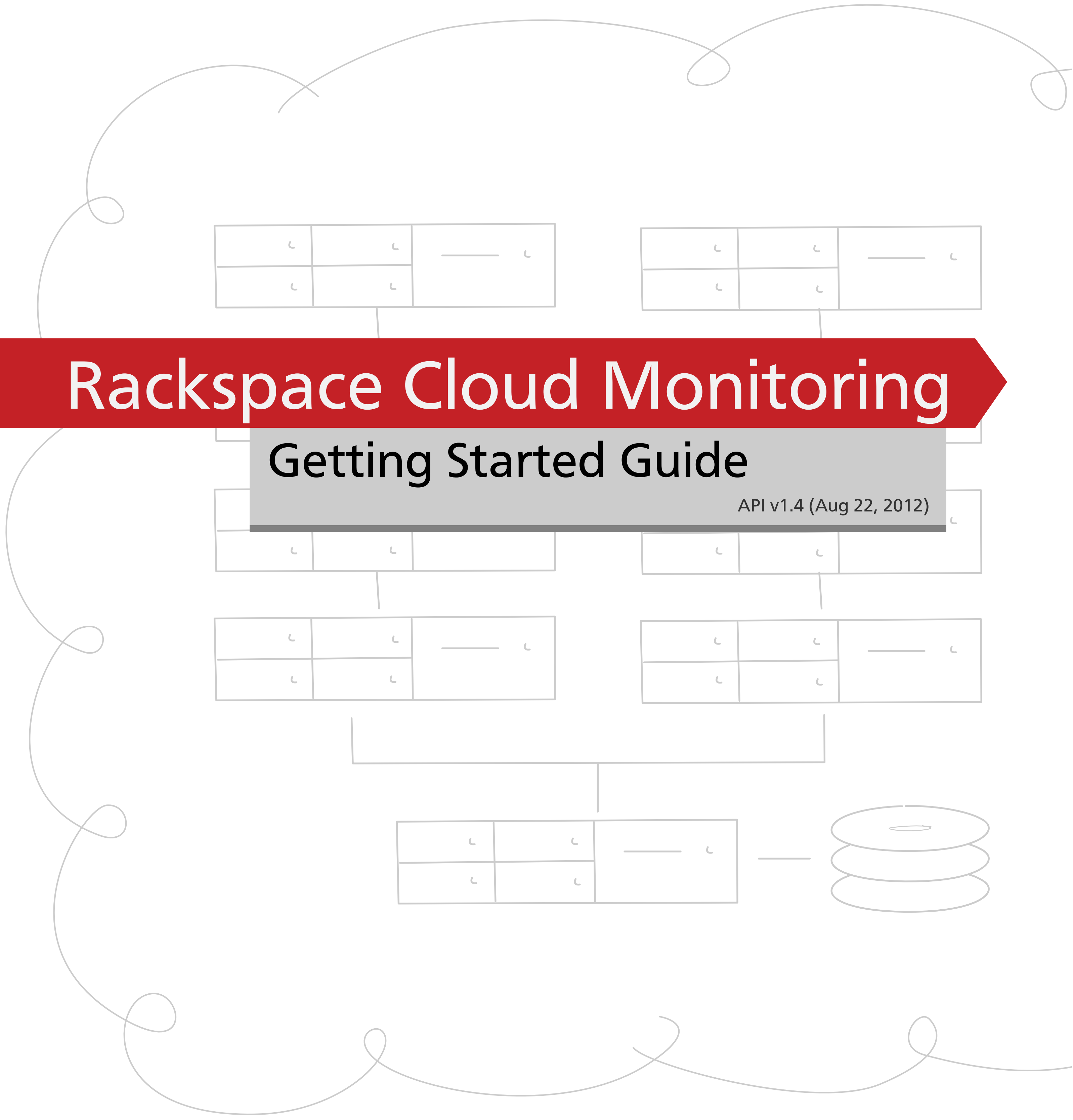


Rackspace Cloud Monitoring

Getting Started Guide

API v1.4 (Aug 22, 2012)

rackspace® HOSTING

docs.rackspace.com/api

Rackspace Cloud Monitoring Getting Started Guide

API v1.4 (2012-08-22)

This document provides in introduction for software developers interested in developing applications using the Rackspace Cloud Monitoring Application Programming Interface (API). The document is for informational purposes only and is provided “AS IS.”

Table of Contents

List of Figures

List of Examples

1. Introduction

When your cloud resources are working as expected, everything is running smoothly and it's business as usual. However, things don't always work perfectly, and problems can occur. When something unexpectedly breaks, it can impact performance, or worse, it can take down your server or company website and prevent you and your customers from accessing critical data. If this happens, you'll want to know right away and take the appropriate actions to keep your business operational and your customers happy. But don't worry, Rackspace Cloud Monitoring can help you by providing timely and accurate information on just how your resources are performing. It supplies you with four key pieces of information that can help you manage your business:

- Current system health
- Alerts on failure conditions
- Collection of historical data
- Trending and capacity planning
- Change here BETS

Using Rackspace Cloud Monitoring's RESTful API you can quickly create multiple monitors using predefined checks, such as PING, HTTPS, SMTP (and many more), to keep track of your cloud resources and receive instant notification when a resource needs your attention.

This guide explains the basics of how to access and use the API so you can begin to monitor your resources immediately.

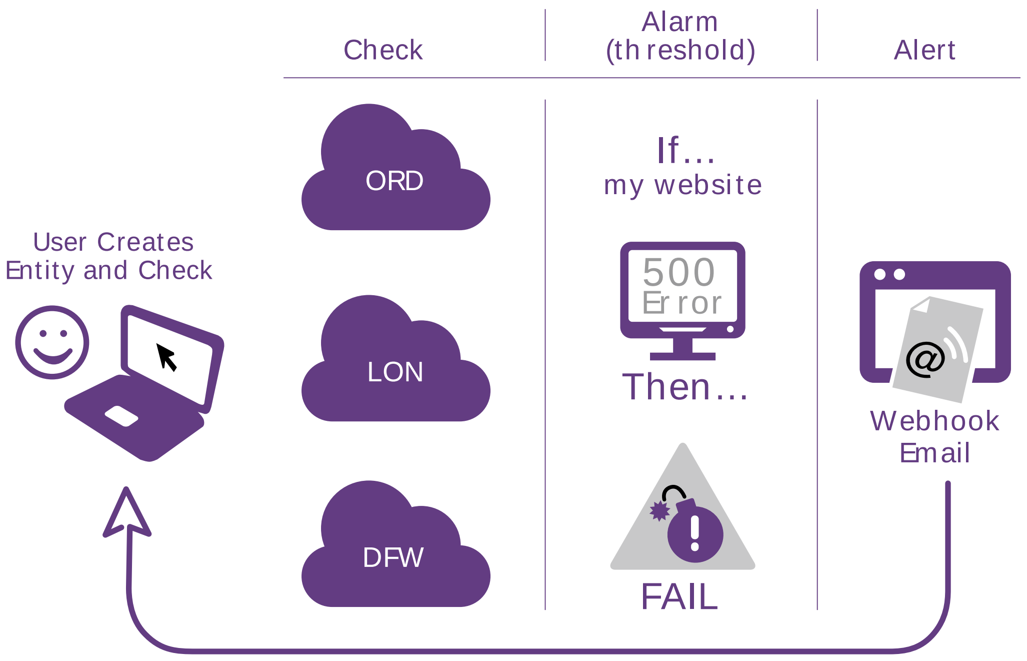
1.1. How Rackspace Cloud Monitoring Works

Rackspace Cloud Monitoring helps you to stay one step ahead of your customers by keeping a keen eye on all of your resources; from web sites to web servers, routers, load balancers, and more.

All you need to do is create an entity that represents the thing you want to monitor and then attach a check to it using one our predefined checks. For example, you might want to use the PING check to monitor your web site's public IP address.

You can run your checks from multiple monitoring zones. Each check has an alarm associated with it that serves as a threshold and processes the output of the check. When a specific condition is met, the alarm is triggered and your notification plan is put into action; sending you an email notification or a webhook to a URL.

Figure 1.1. Rackspace Cloud Monitoring Work Flow



1.2. Prerequisites

Before you attempt to use the Rackspace Cloud Monitoring API, make sure you have met the following prerequisites:

- You have a Rackspace account.
- You are familiar with HTTP v1.1 and *RESTful* web services.

1.3. Accessing the API

To access the Rackspace Cloud Monitoring use the following URL:

https://monitoring.api.rackspacecloud.com/v1.0/*1234*

Note

1234 represents your account number and is used as a placeholder in the examples and exercises in this guide. When you work through the examples you'll need to replace 1234 with your Rackspace account number. If you don't know your account number now, don't worry. We'll explain how to find it later. To see find that information now, read `publicURL` at Section 1.4.4, "Authentication Response Description" [6].

1.4. Authentication

Authentication validates your user credentials and provides access to Rackspace services and products. You'll need to authenticate prior to each request you make to the API. This section gives you a quick introduction to Rackspace authentication so that you can access the Rackspace Cloud Monitoring API. For a complete discussion of authentication, see Cloud Identity Client Developer Guide.

1.4.1. The Authentication Endpoint

You'll send your authentication request to the Rackspace Cloud Authentication Service using the following URL:

https://auth.api.rackspacecloud.com/v1.1/auth

1.4.2. The Authentication Process

When sending an authentication request, you'll send your Rackspace account username and API access key to the Rackspace Cloud Authentication Service. After receiving your request, the Authentication Service validates your credentials and returns an authorization token and a list of the services you can access along with your account number. You'll need to include the authentication token and your account number with each request you make to the API. We'll show you how in Section 1.4.3, "Example Authentication" [4].

If you need help finding your API key, follow the steps below.

Procedure 1.1. To find your API key

1. Log into the Control Panel at https://manage.rackspacecloud.com.
2. When the console opens, click **Your Account**, and then **API Access**.
3. From the API Access page you can generate a new key or **Show/Hide** an existing key.

When you're ready to send your authentication request, you'll need to copy and paste your key into the `key` attribute of the `credentials` parameter, like this:

```
<credentials xmlns="http://docs.rackspacecloud.com/auth/api/v1.1"
              username="your_Rackspace_username"
              key="00000000-0000-0000-0000-000000000000"/>'
```

1.4.3. Example Authentication

The following example demonstrates how to authenticate. Remember to insert your Rackspace user name and API key in the `username` and `key` options.

Example 1.1. Authentication Request: XML

```
curl -i \
-H "Content-Type: application/xml" \
-H "Accept: application/xml" \
-d \
'<?xml version="1.0" encoding="UTF-8"?>
 <credentials xmlns="http://docs.rackspacecloud.com/auth/api/v1.1"
              username="your_Rackspace_username"
              key="00000000-0000-0000-0000-000000000000"/>' \
https://auth.api.rackspacecloud.com/v1.1/auth
```

Example 1.2. Authentication Request: JSON

```
curl -i \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d \
'{
    "credentials": {
    "username": "my_Rackspace_username",
    "key": "00000000-0000-0000-0000-000000000000"}
    }' \
https://auth.api.rackspacecloud.com/v1.1/auth
```

Example 1.3. Authentication Response: XML

```
HTTP/1.1 200 OK
Server: Apache/2.2.3 (Red Hat)
vary: Accept,Accept-Encoding
Content-Type: application/xml
Date: Wed, 22 Aug 2012 23:25:49 GMT
Connection: Keep-Alive
Content-Length: 1901

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<auth xmlns="http://docs.rackspacecloud.com/auth/api/v1.1">
```

```
 <token id="asdasdasd-adsasdads-asdasdasd-adsadsasd" expires=
"2012-08-22T20:09:05.000-05:00"/>

  <serviceCatalog>
    <service name="cloudFilesCDN"><endpoint region="DFW"
 v1Default="true" publicURL="https://cdn1.clouddrive.com/v1/
MossoCloudFS_9c24e3db-52bf-4f26-8dc1-220871796e9f"/></service>
    <service name="cloudFiles"><endpoint region="DFW"
 v1Default="true" publicURL="https://storage101.dfw1.clouddrive.
com/v1/MossoCloudFS_9c24e3db-52bf-4f26-8dc1-220871796e9f"
 internalURL="https://snet-storage101.dfw1.clouddrive.com/v1/
MossoCloudFS_9c24e3db-52bf-4f26-8dc1-220871796e9f"/></service>
    <service name="cloudServers"><endpoint v1Default="true" publicURL="https:/
/servers.api.rackspacecloud.com/v1.0/1234"/></service>
    <service name="cloudServersOpenStack"><endpoint region="DFW" v1Default=
"false" publicURL="https://dfw.servers.api.rackspacecloud.com/v2/661145"/
><endpoint region="ORD" v1Default="false" publicURL="https://ord.servers.api.
rackspacecloud.com/v2/1234"/></service>
    <service name="cloudDNS"><endpoint v1Default="false" publicURL="https://
dns.api.rackspacecloud.com/v1.0/1234"/></service>
    <service name="cloudDatabases"><endpoint region="DFW" v1Default="false"
 publicURL="https://dfw.databases.api.rackspacecloud.com/v1.0/661145"/
><endpoint region="ORD" v1Default="false" publicURL="https://ord.databases.
api.rackspacecloud.com/v1.0/1234"/></service>
    <service name="cloudLoadBalancers"><endpoint region="DFW" v1Default=
"false" publicURL="https://dfw.loadbalancers.api.rackspacecloud.com/v1.
0/1234"/><endpoint region="ORD" v1Default="false" publicURL="https://ord.
loadbalancers.api.rackspacecloud.com/v1.0/1234"/></service>
    <service name="cloudMonitoring"><endpoint v1Default="false" publicURL=
"https://monitoring.api.rackspacecloud.com/v1.0/1234"/></service>
  </serviceCatalog>
</auth>
```

Example 1.4. Authentication Response: JSON

```
HTTP/1.1 200 OK
Server: Apache/2.2.3 (Red Hat)
vary: Accept,Accept-Encoding
Content-Type: application/json
Date: Thu, 23 Aug 2012 00:29:08 GMT
Connection: Keep-Alive
Content-Length: 1451

{
  "auth":{
    "token"{
  "id":"asdasdasd-adsasdads-asdasdasd-adsadsasd",
     "expires":"2012-08-23T19:59:03.512-05:00"},
    "serviceCatalog":{
        "cloudServersOpenStack":[
        {
          "region":"DFW",
          "publicURL":"https:\/\/dfw.servers.api.rackspacecloud.com\/v2\/
1234"},
        {
          "region":"ORD",
          "publicURL":"https:\/\/ord.servers.api.rackspacecloud.com\/v2\/
1234"}],
        "cloudDNS":[
        {
          "publicURL":"https:\/\/dns.api.rackspacecloud.com\/v1.0\/1234"}],
        "cloudFilesCDN":[
        {
          "region":"DFW",
          "publicURL":"https:\/\/cdn1.clouddrive.com\/v1\/
MossoCloudFS_9c24e3db-52bf-4f26-8dc1-220871796e9f","v1Default":true}],
        "cloudFiles":[
        {
        "region":"DFW","publicURL":"https:\/\/storage101.dfw1.clouddrive.com
\/v1\/MossoCloudFS_9c24e3db-52bf-4f26-8dc1-220871796e9f","v1Default":true,
"internalURL":"https:\/\/snet-storage101.dfw1.clouddrive.com\/v1\/
MossoCloudFS_9c24e3db-52bf-4f26-8dc1-220871796e9f"}],
        "cloudLoadBalancers":[
        {
          "region":"ORD","publicURL":"https:\/\/ord.loadbalancers.api.
rackspacecloud.com\/v1.0\/1234"},
        {
          "region":"DFW","publicURL":"https:\/\/dfw.loadbalancers.api.
rackspacecloud.com\/v1.0\/1234"}],
        "cloudMonitoring":[
        {
         "publicURL":"https:\/\/monitoring.api.rackspacecloud.com\/v1.0\/
1234"}],
         "cloudDatabases":[
         {
         "region":"DFW","publicURL":"https:\/\/dfw.databases.api.
rackspacecloud.com\/v1.0\/1234"},
         {
         "region":"ORD","publicURL":"https:\/\/ord.databases.api.
rackspacecloud.com\/v1.0\/1234"}],
         "cloudServers":[
         {
          "publicURL":"https:\/\/servers.api.rackspacecloud.com\/v1.0\/12345",
"v1Default":true}]
         }
       }
}
```

1.4.4. Authentication Response Description

The authentication response contains the following information:

`token id`	Specifies the authentication token. Tokens are valid for a finite duration; a token's default lifespan is twenty-four hours.
`expires`	Denotes the time after which the token will automatically become invalid. A token may be manually revoked before the time identified by the `expires` attribute; this attribute predicts a token's maximum possible lifespan but does not guarantee that it will reach that lifespan. Clients are encouraged to cache a token until it expires.
`serviceCatalog`	Lists the regions and public and private URLs for each of the Rackspace products and services you have access to.
`publicURL`	Lists the public URL for each product and service.

Note

You will find your account number after the final '/' in the `publicURL` field. In this example the account number is 1234.

1.5. Working with the Exercises in this Guide

The main purpose of this guide is to get you up and running quickly with Rackspace Cloud Monitoring. To help you accomplish this goal, we've included a tutorial that lets you try out the basic operations for creating a monitor. To run through the exercises you can use the cURL command line tool or `raxmon`, the Rackspace Cloud Monitoring command line interface. Read on for a brief introduction to both of these tools.

1.5.1. Using cURL

The cURL command line makes it easier to interact with RESTful APIs. It lets you transmit and receive HTTP requests and responses from the command line or from within a shell script. It is available on most UNIX®-like environments, on Mac OS X®, and Windows®. For more information on cURL, visit http://curl.haxx.se.

This guide uses the following cURL command line options:

- `-i`

  Includes the HTTP header in the output.

- `--data-binary`

  Sends the specified data in a POST request to the HTTP server.

- `-H HEADER`

  Specifies an HTTP header in the request.

1.5.1.1. Copying Request Examples

To avoid a lot of typing, you can copy, edit, and paste the cURL request examples from this guide into a terminal or command window.

Procedure 1.2. To copy and paste the cURL request examples

1. Copy and paste the example request into your favorite text editor.

2. Replace `auth_token` with your authorization token and *1234* with your account number.

3. Paste the revised example into a terminal or command window and press **Enter** to execute the command.

1.5.1.2. Escaping Carriage Returns

To make the cURL commands easier to read, the examples have carriage returns at the end of each line, followed by a backslash ('\') to avoid prematurely terminating the command when it's executed. Note that you shouldn't escape carriage returns inside the message body.

In the following XML example, you can see that the lines that are part of the cURL command syntax have all been escaped with a backslash ('\') to indicate that the command continues on the next line:

Example 1.5. Escaping Carriage Returns in cURL

```
curl -i -d \
'<?xml version="1.0" encoding="UTF-8"?>
<credentials xmlns="http://docs.rackspacecloud.com/auth/api/v1.1"
             username="your_username"
             key="your_api_key"/>' \
-H "Content-Type: application/xml" \
-H "Accept: application/xml" \
'https://auth.api.rackspacecloud.com/v1.1/auth'
```

However, the lines *within* the body of the message are *not* escaped with a backslash ('\') to avoid issues with processing the command:

```
'<?xml version="1.0" encoding="UTF-8"?>
<credentials xmlns="http://docs.rackspacecloud.com/auth/api/v1.1"
             username="your_username"
             key="your_api_key"/>' \
```

1.5.2. Using the raxmon Command Line Interface

The `raxmon` command line interface lets you interact with the Rackspace Cloud Monitoring API in a quick and efficient manner. If you're not a programmer by nature, we suggest using `raxmon` to become familiar with the product and its capabilities. Even if you're a programming guru, we think you'll like the speed at which you can set up monitors using `raxmon`. To try out `raxmon`, go to https://github.com/racker/rackspace-monitoring-cli for the installation and configuration information.

Once you have installed `raxmon`, type the following command in a terminal or command window to see a list of the available `raxmon` commands:

`raxmon --help` or `raxmon --h`

To get help for a particular command, type:

`raxmon-command-name --help` or `raxmon-command-name --h`

For example, you can see a brief description of each option you would use to create an entity in the monitoring system by typing the following command:

`raxmon-entities-create --h`

2. Create Your First Monitor

This chapter contains some simple exercises that will help you become familiar with basic monitoring operations. Examples are provided in cURL with JSON formatting. You can also complete this tutorial using the Rackspace Cloud Monitoring command line interface. If you would rather use the command line interface, an example **raxmon** command is also given for each operation. For information on installing and configuring **raxmon** https://github.com/racker/rackspace-monitoring-cli.

For the purpose of this tutorial assume that you have a new web server that you want to make sure is running and responding to requests within a reasonable amount of time. Using this example, you'll perform the following tasks:

1. Create an *entity* to represent the server in the monitoring system.
2. Review the list of available *monitoring zones*.
3. Define three *checks* for the new entity.
4. Set up an email *notification* and a *notification plan* so you can receive information about the entity.
5. Define two *alarms* and assign them to a check to begin the monitoring process.
6. Make a simple modification to the original entity.
7. And finally, you'll delete the entity and it's child objects–the checks and alarms.

Note

When performing the following exercises, you'll need to include your authorization token where you see "`auth_token`" and your account number where you see *1234*. If you haven't requested an authentication token yet, follow the instructions in Example Authentication and then go to Section 2.1, Create an Entity.

2.1. Create an Entity

The first thing you'll do to begin monitoring a resource is to create an entity that represents the resource in the monitoring system.

You'll use the following attributes to create the entity:

`label`	Assigns a meaningful name to the entity. In the examples below, we've named the server "My Rackspace Server". You can choose a different name or use the same one, but note that it's commonly a server name.
`ip_address`	Specifies the server's IP address(es). In the example below, we're specified the IP address and named it "default."

Note

Remember to insert your authorization token and account number as previously described in Section 1.4.3, “Example Authentication” [4].

Example 2.1. Create Entity Request JSON

```
curl -i -X POST \
--data-binary \
'{ "ip_addresses" : { "default" : "192.0.2.15" },
  "label" : "My Rackspace Server",
  "metadata" : {  }
}' \
-H 'X-Auth-Token: auth_token' \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/'
```

Example 2.2. Create Entity Response JSON

```
HTTP/1.1 201 Created
Content-Length: 0
X-Response-Id: .rh-ZWac.h-lon3-maas-prod-api0.r-To4aIzNP.c-25577.
ts-1326394448414.v-0c7bb08
X-Powered-By: Express
Location: https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/
enn14Ch5mc
Date: Thu, 12 Jan 2012 18:54:08 GMT
Content-Type: text/plain
```

If the entity is successfully created, the endpoint returns a response code of 201 Created and a `Location:` header with the URI of the entity. Note that the entity ID is located at the end of the URL. In this example, the entity ID is enn14Ch5mc. Every entity has a unique ID, so yours will be different.

Example 2.3. Create Entity Request using raxmon

```
raxmon-entities-create --label="Monitor Test" --ip-address=default=192.0.2.15
```

Example 2.4. Create Entity raxmon Response

```
HTTP/1.1 201 Created
Content-Length: 0
X-Response-Id: .rh-ZWac.h-lon3-maas-prod-api0.r-To4aIzNP.c-25577.
ts-1326394448414.v-0c7bb08
X-Powered-By: Express
Location: https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/
enn14Ch5mc
Date: Thu, 12 Jan 2012 18:54:08 GMT
Content-Type: text/plain
```

If an error message is returned, the endpoint was unable to create the entity. See error codes descriptions in the Rackspace Cloud Monitoring Developers Guide.

Note

For the remaining steps of this tutorial substitute the ID of the entity you created for "enn14Ch5mc".

Tip

You can request a list of all entities for your account at any time. You can also list checks, alarms, notifications, and so on.

Example 2.5. List All Entities Request using cURL

```
curl -i -X GET \
-H 'X-Auth-Token: auth_token' \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities'
```

Example 2.6. List All Entities using raxmon

```
raxmon-entities-list --details
```

For a complete list of GET commands, see the Quick Reference table in Rackspace Cloud Monitoring Developers Guide

NEXT UP: Check out the monitoring zones you can launch your check from.

2.2. List Monitoring Zones

Before creating a check for the new entity, you'll want to choose which monitoring zones to run the check from. Rackspace Cloud Monitoring is divided into different monitoring zones. You may want to have a single server monitored from several monitoring zones to reduce the risk of false alarms and check the response time from different locations around the world.

Examine the response and choose the monitoring zone(s) you want to launch your first check from.

Example 2.7. List Monitoring Zones using cURL

```
curl -i -X GET \
-H 'X-Auth-Token: auth_token' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/monitoring_zones'
```

Example 2.8. List Monitoring Zones cURL Response

```
HTTP/1.1 200 OK
X-Ratelimit-Remaining: 49982
X-Response-Id: .rh-YQzc.h-lon3-maas-prod-api0.r-kCU9r8nq.c-27354.
ts-1329332815801.v-3aec925
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Lb: lon3-maas-prod-api1
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Date: Wed, 15 Feb 2012 19:06:55 GMT
X-Ratelimit-Window: 24 hours
Content-Type: application/json; charset=UTF-8
```

```
428
{
    "values": [
        {
            "id": "mzdfw",
            "label": "dfw",
            "country_code": "US",
            "source_ips": [
                "2001:4800:7902:0001::/64",
                "50.56.142.128/26"
            ]
        },
        {
            "id": "mzhkg",
            "label": "hkg",
            "country_code": "HK",
            "source_ips": [
                "180.150.149.64/26",
                "2401:1800:7902:1:0:0:0:0/64"
            ]
        },
        {
            "id": "mzlon",
            "label": "lon",
            "country_code": "GB",
            "source_ips": [
                "2a00:1a48:7902:0001::/64",
                "78.136.44.0/26"
            ]
        },
        {
            "id": "mzord",
            "label": "ord",
            "country_code": "US",
            "source_ips": [
                "2001:4801:7902:0001::/64",
                "50.57.61.0/26"
            ]
        }
    ],
    "metadata": {
        "count": 4,
        "limit": 100,
        "marker": null,
        "next_href": null
    }
}
```

Example 2.9. List Monitoring Zones using raxmon

```
raxmon-monitoring-zones-list --details
```

Example 2.10. List Monitoring Zones raxmon Response

```
{'country_code': u'US',
 'driver': <rackspace_monitoring.drivers.rackspace.RackspaceMonitoringDriver
 object at 0x100562410>,
 'extra': {},
 'id': u'mzdfw',
 'label': u'dfw',
 'source_ips': [u'2001:4800:7902:0001::/64', u'50.56.142.128/26']}
{'country_code': u'HK',
 'driver': <rackspace_monitoring.drivers.rackspace.RackspaceMonitoringDriver
 object at 0x100562410>,
 'extra': {},
 'id': u'mzhkg',
 'label': u'hkg',
 'source_ips': [u'180.150.149.64/26', u'2401:1800:7902:1:0:0:0:0/64']}
{'country_code': u'GB',
 'driver': <rackspace_monitoring.drivers.rackspace.RackspaceMonitoringDriver
 object at 0x100562410>,
 'extra': {},
 'id': u'mzlon',
 'label': u'lon',
 'source_ips': [u'2a00:1a48:7902:0001::/64', u'78.136.44.0/26']}
{'country_code': u'US',
 'driver': <rackspace_monitoring.drivers.rackspace.RackspaceMonitoringDriver
 object at 0x100562410>,
 'extra': {},
 'id': u'mzord',
 'label': u'ord',
 'source_ips': [u'2001:4801:7902:0001::/64', u'50.57.61.0/26']}
```

NEXT UP: Create several checks for the new entity.

2.3. Create Checks

2.3.1. Create a PING Check

Any entity that you create can have a multitude of checks, each monitoring a different aspect of the entity. And since you can monitor many different aspects of a single entity, you'll practice creating several new checks in this exercise.

First, you'll create a PING check to verify the web server is responding to the following attributes to create the ping check:

label	Assigns a meaningful name to the check. In the examples below, we've named the check "Website check 1". You can choose a different name or use the same one.
type	Specifies the type of check you're creating.
monitoring_zones_poll	Specifies the monitoring zones that will launch the check. In this example we'll use "mzdfw".
timeout	Specifies the timeout in seconds for the check. This has to be less than the period.
period	Specifies the period in seconds for the check. This specifies how often Rackspace Cloud Monitoring *collectors* run this check.
target_alias	Resolves the check to an IP address.

Example 2.11. Create a PING Check using cURL

```
curl -i \
--data-binary \
'{ "details" : {  },
  "label" : "Website check 1",
  "monitoring_zones_poll" : [ "mzdfw" ],
  "period" : "60",
  "target_alias" : "default",
  "timeout" : 30,
  "type" : "remote.ping"
}' \
-H "X-Auth-Token: auth_token" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14Ch5mc/
checks'
```

Example 2.12. Create Ping Check cURL Response

```
HTTP/1.1 201 Created
Date: Fri, 24 Feb 2012 06:28:51 GMT
Location: https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/
enn14Ch5mc/checks/chyYWNw59I
X-RateLimit-Limit: 50000
X-RateLimit-Remaining: 49969
X-RateLimit-Window: 24 hours
X-RateLimit-Type: global
X-Response-Id: .rh-9pAY.h-lon3-maas-prod-api1.r-2ruPuxLu.c-99739.
ts-1330064931513.v-c576983
X-LB: lon3-maas-prod-api1
Content-Length: 0
Content-Type: text/plain
```

If the check is successfully created, the endpoint returns a response code of 201 and a `Location:` Header containing the URL of the check. In this example, the check id is chyYWNw59I, but yours will be different.

If an error message is returned, the monitoring system was unable to create the check. For information about errors, see the Rackspace Cloud Monitoring Developers Guide.

Note

Checks *always* have a parent entity associated with them. Therefore all URLs are contained underneath the check URL. For example, if the ID of the entity we created earlier is enn14Ch5mc, then the URLs would be underneath https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14Ch5mc/.

Example 2.13. Create a PING Check using raxmon

```
raxmon-checks-create --entity-id=enn14Ch5mc --type=remote.ping --label=
"Website check 1" --monitoring-zones=mzdfw --timeout=30 --period=60 --target-
alias=default
```

Example 2.14. Create a PING Check raxmon Response

```
Resource created. ID: chyYWNw59I
```

NEXT UP: Test the new check to make sure it works.

2.3.2. Test the Check

Testing the check runs the check once and lists the check's metrics. This is an easy way to verify and view your metrics. Later, you can use the test check output to help you build alarms.

Example 2.15. Create Test Check Request using cURL

```
curl -i \
--data-binary \
'{ "details" : {  },
  "label" : "Website check 1",
  "monitoring_zones_poll" : [ "mzdfw" ],
  "period" : "60",
  "target_alias" : "default",
  "timeout" : 30,
  "type" : "remote.ping"
}' \
-H 'X-Auth-Token: auth_code' \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14Ch5mc/test-
check/'
```

Example 2.16. Create Test Check cURL Response

```
HTTP/1.1 200 OK
X-Ratelimit-Remaining: 497
X-Response-Id: .rh-VE2m.h-lon3-maas-prod-api0.r-pm6oiwjm.c-37726.
ts-1329263604920.v-3aec925
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Lb: lon3-maas-prod-api1
X-Ratelimit-Type: test_check
X-Ratelimit-Limit: 500
Date: Tue, 14 Feb 2012 23:53:24 GMT
X-Ratelimit-Window: 24 hours
Content-Type: application/json; charset=UTF-8

[
    {
        "timestamp": 1329263613579,
        "monitoring_zone_id": "mzdfw",
        "available": true,
        "status": "cnt=5,avail=100,min=0.0018,max=0.0020,avg=0.0019",
        "metrics": {
            "minimum": {
                "type": "n",
                "data": "1.808000029996e-03"
            },
            "available": {
                "type": "n",
                "data": "1.000000000000e+02"
            },
            "maximum": {
                "type": "n",
                "data": "1.990000018850e-03"
            },
            "count": {
                "type": "i",
                "data": "5"
            },
            "average": {
                "type": "n",
                "data": "1.866600010544e-03"
            }
        }
    }
]
```

Example 2.17. Test Check Request using raxmon

```
raxmon-checks-test --entity-id=enn14Ch5mc --type=remote.ping --monitoring-
zones=mzdfw --timeout=30 --period=60 --target-alias=default
```

Example 2.18. Test Check raxmon Response

```
[{u'available': True, u'timestamp': 1329334696399,
u'monitoring_zone_id': u'mzdfw',
u'status': u'cnt=5,avail=100,min=0.0018,max=0.0020,avg=0.0019',
u'metrics': {u'count': {u'data': u'5', u'type': u'i'},
u'available': {u'data': u'1.000000000000e+02', u'type': u'n'},
u'average': {u'data': u'1.874800003134e-03', u'type': u'n'},
u'minimum': {u'data': u'1.803999999538e-03', u'type': u'n'},
u'maximum': {u'data': u'2.022000029683e-03', u'type': u'n'}}}]
```

NEXT UP: A PING check doesn't tell you if your web server is running, it only tells you that the server is up. So, let's create an HTTP check to monitor the website directly.

2.3.3. Create HTTP Checks

The HTTP check attempts to retrieve the given URL from the server's IP address. Since an HTTP server might host pages for multiple, different domain names on a single IP address, the check needs a full URL to know which server name and which URL to examine. Likewise, a heavy traffic site might have multiple HTTP servers for the same domain name. The HTTP check lets you monitor each individual web server.

Example 2.19. Create HTTP Check Request using cURL

```
curl -i -X POST \
--data-binary \
{ "details" : { "body" : "foo",
      "method" : "GET",
      "url" : "www.examples.org"
    },
  "label" : "Website check 1",
  "monitoring_zones_poll" : [ "mzdfw" ],
  "period" : "60",
  "target_alias" : "default",
  "type" : "remote.http"
} \
-H 'X-Auth-Token: auth_token' \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14Ch5mc/
checks'
```

Example 2.20. Create HTTP Check cURL Response

```
HTTP/1.1 201 Created
X-Ratelimit-Remaining: 49960
X-Response-Id: .rh-vgB4.h-ord1-maas-prod-api1.r-W0REfAoy.c-2650.
ts-1329337042872.v-b9d7626
Content-Length: 0
X-Lb: ord1-maas-prod-api0
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Location: https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/
enn14Ch5mc/checks/chTTslRf7v
Date: Wed, 15 Feb 2012 20:17:22 GMT
X-Ratelimit-Window: 24 hours
Content-Type: text/plain
```

Example 2.21. Create HTTP Check Request using raxmon

```
raxmon-checks-create --entity-id=enn14Ch5mc --type=remote.http --label=
"Website check 1" --monitoring-zones=mzdfw --details=url=www.examples.org,
body=foo,method=GET timeout=30 --period=60 --target-alias=default
```

Example 2.22. Create HTTP Check raxmon Response

```
Resource created. ID: chTTslRf7v
```

NEXT UP: Now let's configure a second HTTP check to monitor a *different* page on the web server.

Example 2.23. Create a Second HTTP Check Request using cURL

```
curl -i -X POST \
--data-binary \
'{ "details" : { "body" : "foo",
      "method" : "GET",
      "url" : "www.examples.org/test/"
    },
  "label" : "RemoteHTTP for Test page",
  "monitoring_zones_poll" : [ "mzdfw" ],
  "period" : "60",
  "target_alias" : "default",
  "timeout" : "30",
  "type" : "remote.http"
}' \
-H 'X-Auth-Token: auth_token' \
-H "Content-Type: application/json" \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14Ch5mc/
checks
```

Example 2.24. Create a Second HTTP Check cURL Response

```
HTTP/1.1 201 Created
X-Ratelimit-Remaining: 49957
X-Response-Id: .rh-pBYi.h-lon3-maas-prod-api0.r-5VLDWtbw.c-4363.
ts-1329346442692.v-b9d7626
Content-Length: 0
X-Lb: lon3-maas-prod-api0
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Location: https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/
enn14Ch5mc/checks/ch4ccHWyyI
Date: Wed, 15 Feb 2012 22:54:02 GMT
X-Ratelimit-Window: 24 hours
Content-Type: text/plain
```

Example 2.25. Create a Second HTTP Check Request using raxmon

```
raxmon-checks-create --entity-id=enn14Ch5mc --type=remote.http --label=
"RemoteHTTP for Test page" --monitoring-zones=mzdfw --details=url=www.
examples.org/test/,body=foo,method=GET --timeout=30 --period=60 --target-
alias=default
```

Example 2.26. Create a Second HTTP Check raxmon Response

```
Resource created. ID: ch4ccHWyyI
```

NEXT UP: Get a list of the checks you've created for your entity.

2.4. List All Checks for the Entity

You can list the checks for a specific entity at any time by doing GET for a specific entity id.

Example 2.27. List All Checks for an Entity using cURL

```
curl -i -X GET \
-H 'X-Auth-Token: auth_token' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14Ch5mc/
checks'
```

Example 2.28. List all Checks cURL Response

```
HTTP/1.1 200 OK
X-Ratelimit-Remaining: 49967
X-Response-Id: .rh-NGRc.h-dfw1-maas-prod-api0.r-fKxxaymj.c-34581.
ts-1329407832952.v-b9d7626
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Lb: dfw1-maas-prod-api1
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Date: Thu, 16 Feb 2012 15:57:12 GMT
X-Ratelimit-Window: 24 hours
Content-Type: application/json; charset=UTF-8

f37
{
    "values": [
        {
            "id": "ch4ccHWyyI",
            "label": "RemoteHTTP for Test page",
            "type": "remote.http",
            "details": {
                "url": "www.foo.com/test/",
                "body": "foo",
                "method": "GET"
            },
            "monitoring_zones_poll": [
                "mzord"
            ],
            "timeout": 30,
            "period": "60",
            "target_alias": "default",
            "target_hostname": null,
            "target_resolver": null,
            "disabled": false,
            "collectors": [
                "coviYlq3jx"
            ],
            "created_at": 1329346442715,
            "updated_at": 1329346442715
        },
        {
            "id": "chyYWNw59I",
            "label": "Website Ping Check",
            "type": "remote.ping",
            "details": {},
            "monitoring_zones_poll": [
                "mzord"
            ],
            "timeout": 30,
            "period": "60",
            "target_alias": "default",
            "target_hostname": null,
            "target_resolver": null,
            "disabled": false,
            "collectors": [
                "co1LKEam0X"
            ],
            "created_at": 1328285429554,
            "updated_at": 1328285429554
        },
        {
            "id": "chTTslRf7v",
            "label": "RemoteHTTP",
            "type": "remote.http",
            "details": {
                "url": "www.examples.org",
                "body": "foo",
                "method": "GET"
            },
            "monitoring_zones_poll": [
                "mzord"
            ],
            "timeout": 30,
            "period": "60",
            "target_alias": "default",
            "target_hostname": null,
            "target_resolver": null,
            "disabled": false,
            "collectors": [
                "coKaALv9Ml"
            ],
            "created_at": 1329337042992,
            "updated_at": 1329337042992
        },
}
```

Example 2.29. List All Checks using raxmon

```
raxmon-checks-list --entity-id=enn14Ch5mc
```

Example 2.30. List All Checks raxmon response

```
<Check: id=ch4ccHWyyI label=RemoteHTTP for Test page...>
<Check: id=chyYWNw59I label=Website Ping Check...>
<Check: id=chTTs1RF7v label=RemoteHTTP...>
```

NEXT UP: With three new checks created, it's now time to set up notifications.

2.5. Set Up Notifications

In most cases you, and perhaps several people on your team, will be interested in multiple alerts. Rackspace Cloud Monitoring lets you set up notification plans that can be shared between multiple alerts. In this exercise, you'll create the notification first, then the notification plan, and then finally the alarms. Note that you can create alarms and notifications in any order you choose, or simply modify an existing record to create a new one. However, to create an a notification plan, you do need to create a notification first.

First step is we will create an email notification, which will be an attribute to the notification plan.

Example 2.31. Create Notification Request using cURL

```
curl -i -X POST \
--data-binary \
   '{ "details" : { "address" : "joe@example.org" },
  "label" : "Alert email 1",
  "type" : "email"
}' \
-H 'X-Auth-Token: auth_token' \
-H 'Content-Type: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/notifications'
```

Example 2.32. Create Notification cURL Response

```
HTTP/1.1 201 Created
X-Ratelimit-Remaining: 49964
X-Response-Id: .rh-00oi.h-ord1-maas-prod-api1.r-WZWtj0Vc.c-5711.
ts-1329408343356.v-b9d7626
Content-Length: 0
X-Lb: ord1-maas-prod-api1
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Location: https://monitoring.api.rackspacecloud.com/v1.0/1234/notifications/
nt2T8WtWte
Date: Thu, 16 Feb 2012 16:05:43 GMT
X-Ratelimit-Window: 24 hours
Content-Type: text/plain
```

Example 2.33. Create Notification Request using raxmon

```
raxmon-notifications-create --label="Alert email 1" --type=email --details=
address=joe@example.org
```

Example 2.34. Create Notification raxmon Response

```
Resource created. ID: nt2T8WtWte
```

If the notification is successfully created, the endpoint returns a response code of 201 and a `Location:` Header containing the URL of the notification. If an error message is returned, the endpoint was unable to create the notification.

NEXT UP: Create a notification plan so you can receive a message when your entity is in different states.

2.5.1. Create a Notification Plan

A notification represents a single action, whereas a notification plan represents a set of actions. Assuming the ID of the notification you created is nt2T8WtWte, here's an example of how you might specify the attributes for a notification plan:

label	Specifies a descriptive name for the notification plan.
warning_state	Specifies a list of notification ids to send when the state is WARNING.
critical_state	Specifies a list of notification ids to send when the state is CRITICAL.
ok_state	Specifies a list of notification ids to send when the state is OK.

In this case, you're configuring the system to send an email when a state changes.

Example 2.35. Create Notification Plan Request using cURL

```
curl -i -X POST \
--data-binary \
'{
      "label": "Notification Plan 1",
      "warning_state": [
          "nt2T8WtWte"
      ],
      "critical_state": [
          "nt2T8WtWte"
      ],
      "ok_state": [
         "nt2T8WtWte"
      ]
}' \
-H 'X-Auth-Token: 6auth_token' \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/notification_plans'
```

Example 2.36. Create Notification Plan cURL Response

```
HTTP/1.1 201 Created
X-Ratelimit-Remaining: 49958
X-Response-Id: .rh-ew99.h-dfw1-maas-prod-api1.r-dRuk75SF.c-35593.
ts-1329410023443.v-b9d7626
Content-Length: 0
X-Lb: dfw1-maas-prod-api0
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Location: https://monitoring.api.rackspacecloud.com/v1.0/1234/
notification_plans/npkmLh5vVk
Date: Thu, 16 Feb 2012 16:33:43 GMT
X-Ratelimit-Window: 24 hours
Content-Type: text/plain
```

Example 2.37. Create Notification Plan Request using raxmon

```
raxmon-notification-plans-create --label="Notification Plan 1" --critical-
state=nt2T8WtWte --warning-state=nt2T8WtWte --ok-state=nt2T8WtWte
```

Example 2.38. Create Notification Plan raxmon Response

```
Resource created. ID: npkmLh5vVk
```

If the endpoint responds with a 201 and a `Location:` Header containing the URL of the new notification plan, the notification plan was created. If the notification plan was not created, an error is returned.

NEXT UP: Create an alarm for the entity and associate it with your notification plan.

2.6. Create an Alarm

In this exercise, you'll set an alarm for the entity. Here's an example alarm that will send you a warning alert if the average PING response time is over 50ms. Remember that alarms always have a parent entity associated with them, so all URLs are contained underneath the alarm URL. In this example, the entity ID is enn14Ch5mc, so we issue the following request to create the alarm and associate it to the entity:

Example 2.39. Create Alarm Request using cURL

```
curl -i -X POST \
--data-binary \
'{
  "check_type": "remote.ping",
  "notification_plan_id": "npkmLh5vVk",
  "criteria": "if (metric[\"duration\"] < 50) { return OK } return WARNING"
}' \
-H 'X-Auth-Token: auth_token' \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14Ch5mc/
alarms'
```

Example 2.40. Create Alarm cURL Response

```
HTTP/1.1 201 Created
X-Ratelimit-Remaining: 49947
X-Response-Id: .rh-ew99.h-dfw1-maas-prod-api1.r-Kti7H0py.c-39599.
ts-1329418764114.v-b9d7626
Content-Length: 0
X-Lb: dfw1-maas-prod-api0
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Location: https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/
enn14Ch5mc/alarms/alIxnPKcZp
Date: Thu, 16 Feb 2012 18:59:24 GMT
X-Ratelimit-Window: 24 hours
Content-Type: text/plain
```

Example 2.41. Create Alarm Request using raxmon

```
raxmon-alarms-create --check-type=remote.ping --criteria="if (metric[\"average
\"] < 50) { return OK } return WARNING" --notification-plan=npkmLh5vVk --
entity-id=enn14Ch5mc
```

Example 2.42. Create Alarm raxmon Response

```
Resource created. ID: alIxnPKcZp
```

If the endpoint returns a response code is 201 and a `Location:` header containing the URL of the new alarm, the alarm was successfully created, otherwise the endpoint returns an error.

Since we're monitoring two items, both PING times and the HTTP response time, let's add a second alarm to go off when the web response time takes longer than 100ms. Here's an example of how you might specify the attributes for this alarm:

Example 2.43. Create an Alarm HTTP Request using cURL

```
curl -i -X POST \
--data-binary \
'{ "check_type" : "remote.http",
  "criteria" : "if (metric[\"code\"] regex \"^[23]..$\") { return OK } return
 WARNING",
  "notification_plan_id" : "npkmLh5vVk"
}' \
-H 'X-Auth-Token: your_auth_key' \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14Ch5mc/
alarms'
```

Example 2.44. Create an Alarm cURL Response

```
HTTP/1.1 201 Created
X-Ratelimit-Remaining: 49946
X-Response-Id: .rh-E79I.h-ord1-maas-prod-api1.r-HDDENoUp.c-106.
ts-1329419675047.v-a037e7a
Content-Length: 0
X-Lb: ord1-maas-prod-api0
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Location: https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/
enn14Ch5mc/alarms/alVwP6z00k
Date: Thu, 16 Feb 2012 19:14:34 GMT
X-Ratelimit-Window: 24 hours
Content-Type: text/plain
```

Example 2.45. Create an Alarm Request using raxmon

```
raxmon-alarms-create --check-type=remote.http --criteria="if (metric[\
"duration\"] < 100) { return OK } return WARNING" --notification-plan=
npkmLh5vVk --entity-id=enn14Ch5mc
```

Example 2.46. Create an Alarm raxmon Response

```
Resource created. ID: alVwP6z00k
```

Note that because alarms are shared between all of the checks on an entity, if you have multiple HTTP URLs, you will receive alerts if any of the checks are lagging.

You can always list the available alarms by retrieving:

Example 2.47. List Alarms Request using cURL

```
curl -i -X GET \
-H 'X-Auth-Token: auth_token' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14Ch5mc/
alarms'
```

Example 2.48. List Alarms cURL Response

```
HTTP/1.1 200 OK
X-Ratelimit-Remaining: 49941
X-Response-Id: .rh-qRGT.h-ord1-maas-prod-api0.r-UH0HesoS.c-404.
ts-1329420284585.v-a037e7a
```

```
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Lb: ord1-maas-prod-api0
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Date: Thu, 16 Feb 2012 19:24:44 GMT
X-Ratelimit-Window: 24 hours
Content-Type: application/json; charset=UTF-8

36e
{
    "values": [
        {
            "id": "alIxnPKcZp",
            "label": null,
            "check_type": "remote.ping",
            "check_id": null,
            "criteria": "if (metric[\"average\"] < 50) { return OK } return
 WARNING",
            "notification_plan_id": "npkmLh5vVk",
            "created_at": 1329418764882,
            "updated_at": 1329418764882
        },
        {
            "id": "alVwP6z00k",
            "label": null,
            "check_type": "remote.http",
            "check_id": null,
            "criteria": "if (metric[\"average\"] < 100) { return OK } return
 WARNING",
            "notification_plan_id": "npkmLh5vVk",
            "created_at": 1329419675727,
            "updated_at": 1329419675727
        }
    ],
    "metadata": {
        "count": 2,
        "limit": 100,
        "marker": null,
        "next_href": null
    }
}
```

Example 2.49. List Alarms Request using raxmon

```
raxmon-alarms-list --entity-id=enn14Ch5mc --details
```

Example 2.50. List Alarms raxmon Response

```
{'criteria': u'if (metric["average"] < 50) { return OK } return WARNING',
 'driver': <rackspace_monitoring.drivers.rackspace.RackspaceMonitoringDriver
 object at 0x100649ad0>,
 'entity_id': u'enn14Ch5mc',
 'id': u'alIxnPKcZp',
 'notification_plan_id': u'npkmLh5vVk',
 'type': u'remote.ping'}
{'criteria': u'if (metric["average"] < 100) { return OK } return WARNING',
 'driver': <rackspace_monitoring.drivers.rackspace.RackspaceMonitoringDriver
 object at 0x100649ad0>,
 'entity_id': u'enn14Ch5mc',
 'id': u'alVwP6z00k',
 'notification_plan_id': u'npkmLh5vVk',
 'type': u'remote.http'}

Total: 2
```

The results show there are now two alarms for the entity.

NEXT UP: Make a modification to the entity.

2.7. Modify an Entity

Let's assume that the IP address for the web server has changed. You can issue a **PUT** request for only those fields that have changed and the change is reflected in all downstream checks.

Example 2.51. Modify an Entity Request using cURL

```
curl -i -X PUT \
--data-binary \
'{
   "ip_addresses": {
      "default": "192.168.0.1"
          }
}' \
-H 'X-Auth-Token: auth_token' \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14Ch5mc'
```

Example 2.52. Modify an Entity cURL Response

```
HTTP/1.1 204 No Content
X-Ratelimit-Remaining: 49953
X-Response-Id: .rh-NGRc.h-dfw1-maas-prod-api0.r-AudSFsQj.c-36911.
ts-1329412867712.v-b9d7626
Content-Length: 0
X-Lb: dfw1-maas-prod-api1
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Location: https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/
enn14Ch5mc
Date: Thu, 16 Feb 2012 17:21:07 GMT
X-Ratelimit-Window: 24 hours
Content-Type: text/plain
```

Example 2.53. Modify an Entity Request using raxmon

```
raxmon-entities-update --id=enn14Ch5mc --ip-addresses=default=192.168.0.1
```

Example 2.54. Modify an Entity raxmon Response

```
Resource has been successfully updated
```

Note that a PUT updates all top level fields. If a top level field contains a hash, the PUT overwrites the entire field. To avoid this, you must specify the complete hash.

NEXT UP: Deleting an entity.

2.8. Delete an Entity

Now that you have completed the exercises in this tutorial, let's remove the entity we created. Entities may not be removed until all of their checks and alarms have been removed. In this exercise, you'll practise removing one of the checks you created, one of the alarms, and then entity.

Note

To quickly delete an entity and its children, use `raxmon--entities-delete`. This command lets you delete the entity and all of its children in a single command.

Example 2.55. Delete Checks Request using cURL

```
curl -i -X \
DELETE \
-H 'X-Auth-Token: auth_token' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14Ch5mc/
checks/chyYWNw59I
```

Example 2.56. Delete Checks cURL Response

```
HTTP/1.1 204 No Content
X-Ratelimit-Remaining: 49993
X-Response-Id: .rh-Fxp3.h-ord1-maas-prod-api0.r-4Fb5IwLs.c-455.
ts-1330554327071.v-50c8c2d
Content-Length: 0
X-Lb: ord1-maas-prod-api1
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Date: Wed, 29 Feb 2012 22:25:27 GMT
X-Ratelimit-Window: 24 hours
Content-Type: text/plain
```

Example 2.57. Delete Alarms Requests using cURL

```
curl -i -X \
DELETE \
-H 'X-Auth-Token: auth_key' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14CH5mc/
alarms/alIxnPKczp
```

Example 2.58. Delete Alarms cURL Response

```
HTTP/1.1 204 No Content
X-Ratelimit-Remaining: 49987
X-Response-Id: .rh-GLyy.h-dfw1-maas-prod-api0.r-2Ulro75y.c-32883.
ts-1330554811888.v-6fee2d2
Content-Length: 0
X-Lb: dfw1-maas-prod-api1
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Date: Wed, 29 Feb 2012 22:33:31 GMT
X-Ratelimit-Window: 24 hours
Content-Type: text/plain
```

Example 2.59. Delete Entity Request using cURL

```
H 'X-Auth-Token: auth_token' \
-H 'Accept: application/json' \
'https://monitoring.api.rackspacecloud.com/v1.0/1234/entities/enn14CH5mc'
```

Example 2.60. Delete Entity cURL Response

```
HTTP/1.1 204 No Content
X-Ratelimit-Remaining: 49984
X-Response-Id: .rh-zyPa.h-dfw1-maas-prod-api0.r-6RuOWE2V.c-236.
ts-1330555655887.v-50c8c2d
Content-Length: 0
X-Lb: dfw1-maas-prod-api0
X-Ratelimit-Type: global
X-Ratelimit-Limit: 50000
Date: Wed, 29 Feb 2012 22:47:35 GMT
X-Ratelimit-Window: 24 hours
Content-Type: text/plain
```

Example 2.61. Delete Entity Request using raxmon

```
raxmon-entities-delete --id=ennCH15mc
```

Example 2.62. Delete Entity raxmon Response

```
Resource deleted
```

This completes the tutorial on creating, modifying, and deleting a monitor using the Rackspace Cloud Monitoring API.

GET MORE INFO

3. Additional Resources

3.1. More on Monitoring

Find out more about Rackspace Cloud Monitoring in these documents:

- Rackspace Cloud Monitoring API Release Notes
- Rackspace Cloud Monitoring Developers Guide

3.2. Talk to Us

Do you have questions about Rackspace Cloud Monitoring? Join us in our chat room at:

- Freenode IRC at #cloudmonitoring
- Or just click the following link:

  http://webchat.freenode.net?channels=cloudmonitoring&uio=d4

3.3. More From Rackspace

- Find API documentation for additional Rackspace products at:

  http://www.docs.rackspacecloud.com/api/
- For technical articles, videos, and more check out the Rackspace Knowledge Center at:

  http://www.rackspace.com/knowledge_center/

Glossary

A

Alarm
: An alarm contains a set of rules that determine when a notification is triggered.

C

Check
: Checks explicitly specify how you want to monitor an entity.

Collector
: A collector collects data from the monitoring zone. The collector is mapped directly to an individual machine or a virtual machine.

E

Entity
: An entity is a resource that you want to monitor. Some examples are a server, a website, or a service.

M

Monitoring Zone
: A monitoring zone is the "launch point" of a check. You can launch checks from multiple monitoring zones.

N

Notification
: A notification is an informational message sent to one or more addresses when an alarm is triggered.

Notification Plan
: A notification plan is a set of notification rules to execute when an alarm is triggered.

R

RESTful
: A type of web service API that uses Representational State Transfer. REST is the architectural style for hypermedia systems used for the World Wide Web.